Oracle® Database

SQL Language Quick Reference

23ai
F47039-12
February 2025

Oracle Database SQL Language Quick Reference, 23ai

F47039-12

Copyright © 2003, 2025, Oracle and/or its affiliates.

Primary Author: Usha Krishnamurthy

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle®, Java, MySQL, and NetSuite are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

Preface

Audience v
Documentation Accessibility v
Related Documents v
Conventions vi

1 SQL Statements

Syntax for SQL Statements 1-1

2 SQL Functions

Syntax for SQL Functions 2-1

3 SQL Expressions

Syntax for SQL Expression Types 3-1

4 SQL Conditions

Syntax for SQL Condition Types 4-1

5 Subclauses

Syntax for Subclauses 5-1

6 Data Types

Overview of Data Types 6-1
Oracle Built-In Data Types 6-2
Oracle-Supplied Data Types 6-5
Converting to Oracle Data Types 6-6

7 Format Models

Overview of Format Models 7-1
- Number Format Models 7-1
  - Number Format Elements 7-1
- Datetime Format Models 7-3
  - Datetime Format Elements 7-3

A SQL*Plus Commands

SQL*Plus Commands A-1

Index

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 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* and *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 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 SQL statements

ADMINISTER KEY MANAGEMENT

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

ALTER ANALYTIC VIEW

```
ALTER ANALYTIC VIEW [ IF EXISTS ] [ schema. ] analytic_view_name
  { RENAME TO new_av_name | COMPILE | alter_add_cache_clause | alter_drop_cache_clause }
```

ALTER ATTRIBUTE DIMENSION

```
ALTER ATTRIBUTE DIMENSION [ IF EXISTS ] [ schema. ]
  attr_dim_name { RENAME TO new_attr_dim_name | COMPILE }
```

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 }   } ]
  [ ONLY TOPLEVEL ]
```

ALTER CLUSTER

```
ALTER CLUSTER [ IF EXISTS ] [ 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
  | prepare_clause
  | drop_mirror_copy
  | lost_write_protection
  | cdb_fleet_clauses
  | property_clause
  | replay_upgrade_clause
  }
```

ALTER DATABASE DICTIONARY

```
ALTER DATABASE DICTIONARY
{    ENCRYPT CREDENTIALS
   | REKEY CREDENTIALS
   | DELETE CREDENTIALS KEY
}
```

ALTER DATABASE LINK

```
ALTER [ SHARED ] [ PUBLIC ] DATABASE LINK [ IF EXISTS ] dblink
  { CONNECT { ( TO user IDENTIFIED BY password [ dblink_authentication ] )
  | WITH credential }
  | 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
    | drop_diskgroup_file_clause
    | convert_redundancy_clause
    | usergroup_clauses
    | user_clauses
    | file_permissions_clause
    | file_owner_clause
    | scrub_clause
    | quotagroup_clauses
    | filegroup_clauses
    }
  | { diskgroup_name [, diskgroup_name ]...
    | ALL
    } { undrop_disk_clause
      | diskgroup_availability
      | enable_disable_volume
      }
}
```

ALTER DOMAIN

```
ALTER [ USECASE ] DOMAIN [ IF EXISTS ][ schema . ] domain_name
  ( ( ADD | MODIFY) DISPLAY display_expression"
   | DROP DISPLAY
   | ( ADD | MODIFY ) ORDER order_expression
   | DROP ORDER  )
   | annotations_clause
```

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 [ IF EXISTS ] [ schema. ] function_name
{ function_compile_clause | { EDITIONABLE | NONEDITIONABLE } }
```

ALTER HIERARCHY

```
ALTER HIERARCHY [ IF EXISTS ] [ schema. ] hierarchy_name
  { RENAME TO new_hier_name | COMPILE }
```

ALTER INDEX

```
ALTER INDEX [ schema.]index_name [ index_ilm_clause ]
  { { deallocate_unused_clause
    | allocate_extent_clause
    | shrink_clause
    | parallel_clause
    | physical_attributes_clause
    | logging_clause
```

```
    | partial_index_clause
    } ...
  | rebuild_clause [ { DEFERRED | IMMEDIATE } INVALIDATION ]
  | PARAMETERS ( 'ODCI_parameters' )
  | COMPILE
  | { ENABLE | DISABLE }
  | UNUSABLE [ ONLINE ] [ { DEFERRED | IMMEDIATE } INVALIDATION ]
  | VISIBLE | INVISIBLE
  | RENAME TO new_name
  | COALESCE [ CLEANUP ] [ ONLY ] [ parallel_clause ]
  | { MONITORING | NOMONITORING } USAGE
  | UPDATE BLOCK REFERENCES
  | alter_index_partitioning
  | annotations_clause
  }
```

ALTER INDEXTYPE

```
ALTER INDEXTYPE [ IF EXISTS ] [ 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 INMEMORY JOIN GROUP

```
ALTER INMEMORY JOIN GROUP [ IF EXISTS ] [ schema. ] join_group
  { ADD | REMOVE } ( [ schema. ] table ( column ) )
```

ALTER JAVA

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

ALTER JSON RELATIONAL DUALITY VIEW

```
ALTER JSON [ RELATIONAL ] DUALITY VIEW [ IF EXISTS ]
  view_name duality_view_replication_clause
```

ALTER LIBRARY

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

ALTER LOCKDOWN PROFILE

```
ALTER LOCKDOWN PROFILE
  { lockdown_features
  | lockdown_options
  | lockdown_statements
  [ USERS = { ALL | COMMON | LOCAL } ]
  }
```

ALTER MATERIALIZED VIEW

```
ALTER MATERIALIZED VIEW [ IF EXISTS ]
  [ schema. ] materialized_view
  [ physical_attributes_clause
```

```
  | modify_mv_column_clause
  | table_compression
  | inmemory_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 ]
  [ { ENABLE | DISABLE } ON QUERY COMPUTATION ]
  [ alter_query_rewrite_clause
   | ( ENABLE | DISABLE ) CONCURRENT REFRESH
   | COMPILE
   | CONSIDER FRESH
  ]
  [ annotations_clause ]
```

ALTER MATERIALIZED VIEW LOG

```
ALTER MATERIALIZED VIEW LOG [ IF EXISTS ] [ 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 [ IF EXISTS ] [ schema. ] zonemap_name
  { alter_zonemap_attributes
  | zonemap_refresh_clause
  | { ENABLE | DISABLE } PRUNING
  | COMPILE
  | REBUILD
  | UNUSABLE
  }
```

ALTER MLE ENV

```
 ALTER MLE ENV [ IF EXISTS ] [schema .] name
  ( ADD IMPORTS ( (import_name MODULE [schema.] mle_module_name)[,(import_name MODULE [schema.]
mle_module_name)... ] )
   | DROP IMPORTS ( (import_name)[,(import_name)...] )
   | ALTER IMPORTS ( (import_name MODULE [schema .] mle_module_name) [,(import_name MODULE
[schema .] mle_module_name)... ] )
   | SET LANGUAGE OPTIONS option_string
   | COMPILE
   )
```

ALTER MLE MODULE

```
ALTER MLE MODULE [ IF EXISTS ] [schema.] module_name
           SET METADATA USING CLOB [() CLOB [])]
```

ALTER OPERATOR

```
ALTER OPERATOR [ IF EXISTS ][ 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 [ IF EXISTS ][ 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
  | application_clauses
  | snapshot_clauses
  | prepare_clause
  | drop_mirror_copy
  | lost_write_protection
  | pdb_managed_recovery
  | [ ENABLE | DISABLE ] BACKUP
  }
```

ALTER PMEM FILESTORE

```
 ALTER PMEM FILESTORE  filestore_name
  (
   ( [ RESIZE size_clause ]  |  autoextend_clause )
  | ( MOUNT [ (MOUNTPOINT file_path | BACKINGFILE file_name) ] [ FORCE ] )
  | DISMOUNT
  )
```

ALTER PROCEDURE

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

ALTER PROFILE

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

ALTER PROPERTY GRAPH

```
ALTER PROPERTY GRAPH [ IF EXISTS ][ schema . ] graph_name COMPILE
```

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 AS ' [ domain_name_of_directory_group | ( AZURE_ROLE = value )
                | IAM_GROUP_NAME = value  ] '
      }
  }
  [ CONTAINER = { CURRENT | ALL } ]
```

ALTER ROLLBACK SEGMENT

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

ALTER SEQUENCE

```
ALTER SEQUENCE [ IF EXISTS ][ schema. ] sequence
  {
    { INCREMENT BY | START WITH } integer
  | { MAXVALUE integer | NOMAXVALUE }
  | { MINVALUE integer | NOMINVALUE }
  |   RESTART
  | { CYCLE | NOCYCLE }
  | { CACHE integer | NOCACHE }
  | { ORDER | NOORDER }
  | { KEEP | NOKEEP }
  | { SCALE {EXTEND | NOEXTEND} | NOSCALE }
  | { SHARD {EXTEND | NOEXTEND} | NOSHARD }
  | { 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
    }
```

```
  | { ENABLE | DISABLE } SHARD DDL
  | SYNC WITH PRIMARY
  | alter_session_set_clause
  }
```

ALTER SYNONYM

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

ALTER SYSTEM

```
ALTER SYSTEM
  { archive_log_clause
  | checkpoint_clause
  | check_datafiles_clause
  | distributed_recov_clauses
  | flush_clause
  | end_session_clauses
  | SWITCH LOGFILE
  | { SUSPEND | RESUME }
  | quiesce_clauses
  | rolling_migration_clauses
  | rolling_patch_clauses
  | security_clauses
  | affinity_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
  | cancel_sql_clause
  }
```

ALTER TABLE

```
ALTER TABLE [ IF EXISTS ][ schema. ] table
  [ memoptimize_read_clause ] [ memoptimize_write_clause ]
  [ alter_table_properties
  | column_clauses
  | constraint_clauses
  | alter_table_partitioning [ { DEFERRED | IMMEDIATE } INVALIDATION ]
  | alter_external_table
  | move_table_clause
  | modify_to_partitioned
  | modify_opaque_type
  | immutable_table_clauses
  | blockchain_table_clauses
  ]
  [ enable_disable_clause
  | { ENABLE | DISABLE }
    { TABLE LOCK | ALL TRIGGERS | CONTAINER_MAP | CONTAINERS_DEFAULT }
  ] ...
```

ALTER TABLESPACE

```
ALTER TABLESPACE [ IF EXISTS ] tablespace alter_tablespace_attrs
```

ALTER TABLESPACE SET

```
ALTER TABLESPACE SET tablespace_set alter_tablespace_attrs
```

ALTER TRIGGER

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

ALTER TYPE

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

ALTER USER

```
ALTER USER [ IF EXISTS ]
  { user
    { IDENTIFIED
      { (BY password [ REPLACE old_password ])
      | ( EXTERNALLY [ AS ' certificate_DN ' | AS ' kerberos_principal_name ' ])
      | ( GLOBALLY [ AS ' [ directory_DN ] | [ {AZURE_USER | AZURE_ROLE} = value ]
      | [ { IAM_GROUP_NAME | IAM_PRINCIPAL_NAME
          | IAM_PRINCIPAL_OCID }  = value ]) '] )
      }
    | ( NO AUTHENTICATION )
    | ( DEFAULT COLLATION collation_name )
    | ( DEFAULT TABLESPACE tablespace )
    | [ LOCAL ] 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 ]
    | [HTTP] DIGEST { ENABLE | DISABLE }
    | CONTAINER = { CURRENT | ALL }
    | { ENABLE | DISABLE } DICTIONARY PROTECTION
    | {( READ ONLY) | (READ WRITE )}
    | container_data_clause
    } ...
  | user [, user ]... proxy_clause
  }
```

ALTER VIEW

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

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 (Unified Auditing)

```
AUDIT
  { POLICY policy
    [ { BY user [, user]... }
    | { EXCEPT user [, user]... }
    | by_users_with_roles ]
    [ 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 ]
  ]
```

CREATE ANALYTIC VIEW

```
CREATE [ OR REPLACE ] [ { FORCE | NOFORCE } ]
   ANALYTIC VIEW [IF NOT EXISTS ] [ schema. ] analytic_view
    [ SHARING = ( METADATA | NONE ) ]
    [ classification_clause ]...
    using_clause
    dim_by_clause
    measures_clause
    [ default_measure_clause ]
    [ default_aggregate_clause ]
    [ cache_clause ]
    [ fact_columns_clause ]
    [ qry_transform_clause ]
```

CREATE ATTRIBUTE DIMENSION

```
CREATE [ OR REPLACE ] [ FORCE | NOFORCE ] ATTRIBUTE DIMENSION
  [IF NOT EXISTS ][ schema. ] attr_dimension
  [ SHARING = ( METADATA | NONE ) ]
  [ classification_clause ]... ]
  [ DIMENSION TYPE { STANDARD | TIME } ]
  attr_dim_using_clause
  attributes_clause
  [ attr_dim_level_clause ]...
  [ all_clause ]
```

CREATE AUDIT POLICY

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

CREATE CLUSTER

```
CREATE CLUSTER [IF NOT EXISTS ][ schema. ] cluster [ SHARING = ( METADATA | NONE ) ]
  (column datatype [ COLLATE column_collation_name ] [ SORT ]
    [, column datatype [ COLLATE column_collation_name ] [ 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
  [ SHARING = ( METADATA | NONE ) ]
  [ 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
  | SET STANDBY LOGGING FOR {DATA AVAILABILITY | LOAD PERFORMANCE}
  ]...
  [ 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 [IF NOT EXISTS ] dblink
  [ CONNECT
    { TO { CURRENT_USER | user IDENTIFIED BY password [ dblink_authentication ] }
         | WITH credential }
    }
  | 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 [IF NOT EXISTS ] directory
  [ SHARING = { METADATA | NONE } ]
  AS 'path_name'
```

CREATE DISKGROUP

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

CREATE DOMAIN

```
{ create_single_column_domain
             | create_multi_column_domain
             | create_flexible_domain  }
```

CREATE SINGLE COLUMN DOMAIN

```
CREATE [ USECASE ] DOMAIN [IF NOT EXISTS ][ schema .] domain_name AS datatype [ STRICT ]
[column_properties_clause]
 [ DISPLAY display_expression ]
 [ ORDER order_expression ]
 [ annotations_clause ]
```

CREATE MULTI COLUMN DOMAIN

```
CREATE [ USECASE ] DOMAIN [ IF NOT EXISTS ][ schema .] domain_name AS
    ( domain_column AS datatype [ STRICT ] [ column_properties_clause ]
     [, domain_column AS datatype [ STRICT ] [ column_properties_clause ] )
   [DISPLAY display_expression ]
   [ORDER order_expression ]
   [annotations_clause ]
```

CREATE FLEXIBLE DOMAIN

```
CREATE [ USECASE ] FLEXIBLE DOMAIN [IF NOT EXISTS ][ schema .]domain_name
    ( domain_column [ , domain_column... ] )
    CHOOSE DOMAIN USING ( domain_discriminant_column datatype)[ , domain_discriminant_column
datatype...] )
FROM
  { DECODE (expr , (search_expr , result_expr) [, search_expr , result_expr ]...[ , default ] )
   | case_expression
  }
```

CREATE EDITION

```
CREATE EDITION [IF NOT EXISTS ] 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 FLEXIBLE DOMAIN

```
CREATE FLEXIBLE DOMAIN [IF NOT EXISTS ][ schema .]domain_name
    ( domain_column [ , domain_column... ] )
    CHOOSE DOMAIN USING ( domain_discriminant_column datatype)[ , domain_discriminant_column
```

```
datatype...] )
  FROM
   { DECODE (expr , search_expr , result_expr [, search_expr , result_expr ]... [ , default ] )
   | case_expression
  }
```

CREATE FUNCTION

```
CREATE [ OR REPLACE ]
[ EDITIONABLE | NONEDITIONABLE ]
FUNCTION [IF NOT EXISTS ] plsql_function_source
```

CREATE HIERARCHY

```
CREATE [ OR REPLACE ] [ FORCE | NOFORCE ]
  HIERARCHY [IF NOT EXISTS ] [ schema. ] hierarchy
  [ SHARING = ( METADATA | NONE ) ]
  [ classification_clause ]... ]
  hier_using_clause
  level_hier_clause
  [ hier_attrs_clause ]
```

CREATE HYBRID VECTOR INDEX

```
 CREATE HYBRID VECTOR INDEX [schema.]index_name ON
  [schema.]table_name(index_column_name)
   PARAMETERS ('paramstring')
```

CREATE INDEX

```
CREATE [ UNIQUE | BITMAP ] INDEX [IF NOT EXISTS ][ schema. ] index_name [ index_ilm_clause ]
  ON { cluster_index_clause
     | table_index_clause
     | bitmap_join_index_clause
     }
[ USABLE | UNUSABLE ]
[ { DEFERRED | IMMEDIATE } INVALIDATION ]
```

CREATE INDEXTYPE

```
CREATE [ OR REPLACE ] INDEXTYPE [IF NOT EXISTS ] [ schema. ] indextype
  [ SHARING = ( METADATA | NONE ) ]
  FOR [ schema. ] operator (parameter_type [, parameter_type ]...)
        [, [ schema. ] operator (parameter_type [, parameter_type ]...)
        ]...
  using_type_clause
  [WITH LOCAL [RANGE] PARTITION ]
  [ storage_table_clause ]
```

CREATE INMEMORY JOIN GROUP

```
CREATE INMEMORY JOIN GROUP [IF NOT EXISTS ][ schema. ] join_group
  ( [ schema. ] table ( column ) , [ schema. ] table ( column )
    [, [ schema. ] table ( column ) ]... )
```

CREATE JAVA

```
CREATE [ OR REPLACE ] [ AND { RESOLVE | COMPILE } ] [ NOFORCE ]
  JAVA [IF NOT EXISTS ] { { SOURCE | RESOURCE } NAMED [ schema. ] primary_name
       | CLASS [ SCHEMA schema ]
       }
  [ SHARING = { METADATA | NONE } ]
  [ 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 JSON RELATIONAL DUALITY VIEW

```
CREATE [ OR REPLACE ]  [ [ NO ] FORCE ] [ EDITIONABLE | NONEDITIONABLE ]
    JSON [ RELATIONAL ] DUALITY VIEW [ IF NOT EXISTS ] view_name
    AS [ duality_view_replication_clause ]
    { { SELECT  object_gen_clause FROM root_table [ root_table_alias ]
   [ table_tags_clause ] } | ( graphql_query_for_DV  ) }
```

CREATE LIBRARY

```
CREATE [ OR REPLACE ]
[ EDITIONABLE | NONEDITIONABLE ]
LIBRARY [IF NOT EXISTS ] plsql_library_source
```

CREATE LOCKDOWN PROFILE

```
CREATE LOGICAL PARTITION TRACKING ON table_name
  PARTITION BY RANGE ( column )...
  [ INTERVAL ( expr ) ]
  ( ( PARTITION [ partition ] range_values_clause )[, PARTITION [ partition ]
range_values_clause ]... )
```

CREATE LOGICAL PARTITION TRACKING

```
CREATE LOCKDOWN PROFILE profile_name
```

CREATE MATERIALIZED VIEW

```
CREATE MATERIALIZED VIEW [IF NOT EXISTS ] [ 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]]
         }
      ]...
    )
  ]
  [ DEFAULT COLLATION collation_name ]
  { 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 ]
  [ evaluation_edition_clause ]
  [ { ENABLE | DISABLE } ON QUERY COMPUTATION ]
  [ query_rewrite_clause ]
  [ { ENABLE | DISABLE } CONCURRENT REFRESH ]
  [ annotations_clause ]
   AS subquery
```

CREATE MATERIALIZED VIEW LOG

```
CREATE MATERIALIZED VIEW LOG [IF NOT EXISTS ] ON [ schema. ] table
  [ SHARING = ( METADATA | NONE ) ]
```

```
  [ 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 MLE ENV

```
CREATE [ OR REPLACE ] MLE ENV [IF NOT EXISTS][schema .] name
   (  [ CLONE [schema .] environment_name  ]
    |
    ( [ IMPORTS ( ( 'import_name' MODULE [schema .] mle_module_name)[,(mle_module_name)...] ) ]
    [ LANGUAGE OPTIONS option_string ] ) )
```

CREATE MLE MODULE

```
CREATE [ OR REPLACE ] MLE MODULE [IF NOT EXISTS][schema .] module_name
 LANGUAGE [schema .] mle_language  [ VERSION version_string ]
 ( USING  BFILE ( directory_object_name , server_file_name )
    | ( CLOB | BLOB | BFILE ) selection_clause
    |  AS module_text )
```

CREATE MULTI COLUMN DOMAIN

```
CREATE DOMAIN [ IF NOT EXISTS ][ schema .] domain_name AS
    ( domain_column AS datatype [ STRICT ] [ column_properties_clause ]
     [, domain_column AS datatype [ STRICT ] [ column_properties_clause ] )
   [DISPLAY display_expression ]
   [ORDER order_expression ]
   [annotations_clause ]
```

CREATE OPERATOR

```
CREATE [ OR REPLACE ] OPERATOR [IF NOT EXISTS ]
   [ schema. ]    operator binding_clause
   [ SHARING = ( METADATA | NONE ) ]
```

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 [IF NOT EXISTS ] plsql_package_source
```

CREATE PACKAGE BODY

```
CREATE [ OR REPLACE ]
[ EDITIONABLE | NONEDITIONABLE ]
PACKAGE BODY [IF NOT EXISTS ] plsql_package_body_source
```

CREATE PFILE

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

CREATE PLUGGABLE DATABASE

```
CREATE PLUGGABLE DATABASE
  { { pdb_name [ AS APPLICATION CONTAINER ] } | { AS SEED } }
  {  create_pdb_from_seed | create_pdb_clone | create_pdb_from_xml | create_pdb_from_mirror_copy
    | using_snapshot_clause | container_map_clause } pdb_snapshot_clause;
```

CREATE PMEM FILESTORE

```
CREATE PMEM FILESTORE filestore_name
  ( (MOUNTPOINT file_path)
  | (BACKINGFILE file_name [ REUSE ])
  | (SIZE size_clause)
  | (BLOCKSIZE size_clause)
  | autoextend_clause
  )
```

CREATE PROCEDURE

```
CREATE [ OR REPLACE ]
[ EDITIONABLE | NONEDITIONABLE ]
PROCEDURE [IF NOT EXISTS ] plsql_procedure_source
```

CREATE PROPERTY GRAPH

```
CREATE [ ( OR REPLACE ) ] PROPERTY GRAPH [IF NOT EXISTS ] [ schema . ] graph_name
     vertex_tables_clause [ edge_tables_clause ]  [ graph_options ]
```

CREATE PROFILE

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

CREATE RESTORE POINT

```
CREATE [ CLEAN ] RESTORE POINT restore_point
   [ FOR PLUGGABLE DATABASE pdb_name ]
   [ AS OF {TIMESTAMP | SCN} expr ]
   [ PRESERVE
   | GUARANTEE FLASHBACK DATABASE
   ]
```

CREATE ROLE

```
CREATE ROLE role
   [ NOT IDENTIFIED
   | IDENTIFIED { BY password
                | USING [ schema. ] package
                | EXTERNALLY
                | GLOBALLY AS '[ domain_name_of_directory_group | ( AZURE_ROLE =
value )
                                | IAM_GROUP_NAME  = value  ]'
                }
   ] [ 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 [IF NOT EXISTS ] [ schema. ] sequence
   [ SHARING = { METADATA | DATA | NONE } ]
   [ { INCREMENT BY | START WITH } integer
   | { MAXVALUE integer | NOMAXVALUE }
   | { MINVALUE integer | NOMINVALUE }
   | { CYCLE | NOCYCLE }
   | { CACHE integer | NOCACHE }
   | { ORDER | NOORDER }
   | { KEEP | NOKEEP }
   | { SCALE {EXTEND | NOEXTEND} | NOSCALE }
   | { SHARD {EXTEND | NOEXTEND} | NOSHARD }
   | { SESSION | GLOBAL }
   ]...
```

CREATE SINGLE COLUMN DOMAIN

```
CREATE DOMAIN [IF NOT EXISTS ][ schema .] domain_name
AS { datatype  |  ENUM ( enum_list ) }
[ STRICT ] [column_properties_clause]
 [ DISPLAY display_expression ]
 [ ORDER order_expression ]
 [ annotations_clause ]
```

CREATE SPFILE

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

CREATE SYNONYM

```
CREATE [ OR REPLACE ] [ EDITIONABLE | NONEDITIONABLE ]
  [ PUBLIC ] SYNONYM [IF NOT EXISTS ]
  [ schema. ] synonym
```

```
  [ SHARING = { METADATA | NONE } ]
  FOR [ schema. ] object [ @ dblink ]
```

CREATE TABLE

```
CREATE [ { GLOBAL | PRIVATE } TEMPORARY | SHARDED | DUPLICATED |
  [ IMMUTABLE ] BLOCKCHAIN | IMMUTABLE ] [ JSON COLLECTION ]
  TABLE [IF NOT EXISTS ] [ schema. ] table
  [ SHARING = { METADATA | DATA | EXTENDED DATA | NONE } ]
  { relational_table | object_table | XMLType_table }
  [ MEMOPTIMIZE FOR READ ]
  [ MEMOPTIMIZE FOR WRITE ]
  [ PARENT [ schema. ] table ] [ MEMOPTIMIZE FOR READ ]
   [ , [ DOMAIN ] [domain_owner.]domain_name (column_name_list) ]
```

CREATE TABLESPACE

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

CREATE TABLESPACE SET

```
CREATE TABLESPACE SET tablespace_set
  [ IN SHARDSPACE shardspace ]
  [ USING TEMPLATE
    ( { DATAFILE [, file_specification ]... ] permanent_tablespace_attrs )
  ]
```

CREATE TRIGGER

```
CREATE [ OR REPLACE ]
[ EDITIONABLE | NONEDITIONABLE ]
TRIGGER [IF NOT EXISTS ] plsql_trigger_source
```

CREATE TYPE

```
CREATE [OR REPLACE]
[ EDITIONABLE | NONEDITIONABLE ]
TYPE [IF NOT EXISTS ] plsql_type_source
```

CREATE TYPE BODY

```
CREATE [ OR REPLACE ]
[ EDITIONABLE | NONEDITIONABLE ]
TYPE BODY [IF NOT EXISTS ] plsql_type_body_source
```

CREATE USER

```
CREATE USER [IF NOT EXISTS ] user
{ {
   IDENTIFIED
        { ( BY password [ [HTTP] DIGEST { ENABLE | DISABLE } ]
        | EXTERNALLY [ AS 'certificate_DN' | AS 'kerberos_principal_name' ]
        | GLOBALLY [ AS '[ directory_DN ] | [ {AZURE_USER | AZURE_ROLE} = value ]
                    | [ IAM_GROUP_NAME | IAM_PRINCIPAL_NAME
                    | IAM_PRINCIPAL_OCID = value ]' }

     | NO AUTHENTICATION
     [ DEFAULT COLLATION collation_name
     | DEFAULT TABLESPACE tablespace
     | [ LOCAL ] 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 }
    | { (READ ONLY) | (READ WRITE) }
    ]...
  ]
```

CREATE VECTOR INDEX

```
CREATE VECTOR INDEX vector_index_name
    ON table_name ( vector_column )
    INCLUDE ( covering_column [,covering_column] )[ GLOBAL ]
    vector_index_organization_clause
    [ WITH TARGET ACCURACY percentage_value ]
    vector_index_parameters_clause
    [ PARALLEL degree_of_parallelism ]
```

CREATE VIEW

```
CREATE [OR REPLACE]
  [[NO] FORCE]
  [ EDITIONING | EDITIONABLE [ EDITIONING ] | NONEDITIONABLE ]
  [ JSON COLLECTION ]
  VIEW [IF NOT EXISTS ] [schema.] view
  [ SHARING = { METADATA | DATA | EXTENDED DATA | NONE } ]
  [ ( { alias [ VISIBLE | INVISIBLE ] [ inline_constraint... ]
      | out_of_line_constraint
      }
        [, { alias [ VISIBLE | INVISIBLE ] [ inline_constraint...]
           | out_of_line_constraint
           }
        ]
    )
  | object_view_clause
  | XMLType_view_clause
  ]
  [ DEFAULT COLLATION collation_name ]
  [ BEQUEATH { CURRENT_USER | DEFINER } ]
  [ annotations_clause ]
  AS subquery [ subquery_restriction_clause ]
  [ CONTAINER_MAP | CONTAINERS_DEFAULT ]
```

DELETE

```
DELETE [ hint ]
   [ FROM ]
   { dml_table_expression_clause
   | ONLY (dml_table_expression_clause)
   } [ t_alias ]
     [ from_using_clause ]
     [ 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 ANALYTIC VIEW

```
DROP ANALYTIC VIEW [ IF EXISTS ][ schema. ] analytic_view_name;
```

DROP ATTRIBUTE DIMENSION

```
DROP ATTRIBUTE DIMENSION [ IF EXISTS ][ schema. ] attr_dimension_name;
```

DROP AUDIT POLICY

```
DROP AUDIT POLICY policy
```

DROP CLUSTER

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

DROP CONTEXT

```
DROP CONTEXT namespace
```

DROP DATABASE

```
DROP DATABASE
```

DROP DATABASE LINK

```
DROP [ PUBLIC ] DATABASE LINK [ IF EXISTS ] dblink
```

DROP DIMENSION

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

DROP DIRECTORY

```
DROP DIRECTORY [ IF EXISTS ] directory_name
```

DROP DISKGROUP

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

DROP DOMAIN

```
DROP [ USECASE ] DOMAIN [IF EXISTS ] [ schema .] domain_name [ FORCE [ PRESERVE ] ]
```

DROP EDITION

```
DROP EDITION [ IF EXISTS ] edition [CASCADE]
```

DROP FLASHBACK ARCHIVE

```
DROP FLASHBACK ARCHIVE flashback_archive;
```

DROP FUNCTION

```
DROP FUNCTION [ IF EXISTS ][ schema. ] function_name
```

DROP HIERARCHY

```
DROP HIERARCHY [ IF EXISTS ][ schema. ] hierarchy_name;
```

DROP INDEX

```
DROP INDEX [ IF EXISTS ][ schema. ] index [ ONLINE ] [ FORCE ] [ { DEFERRED | IMMEDIATE }
INVALIDATION ]
```

DROP INDEXTYPE

```
DROP INDEXTYPE [ IF EXISTS ][ schema. ] indextype [ FORCE ]
```

DROP INMEMORY JOIN GROUP

```
DROP INMEMORY JOIN GROUP [ IF EXISTS ][ schema. ] join_group
```

DROP JAVA

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

DROP LIBRARY

```
DROP LIBRARY [ IF EXISTS ] library_name
```

DROP LOCKDOWN PROFILE

```
DROP LOCKDOWN PROFILE profile_name
```

DROP MATERIALIZED VIEW

```
DROP MATERIALIZED VIEW [ IF EXISTS ] [ schema. ] materialized_view
   [ PRESERVE TABLE ]
```

DROP MATERIALIZED VIEW LOG

```
DROP MATERIALIZED VIEW LOG [ IF EXISTS ] ON [ schema. ] table
```

DROP MATERIALIZED ZONEMAP

```
DROP MATERIALIZED ZONEMAP [ IF EXISTS ] [ schema. ] zonemap_name
```

DROP MLE ENV

```
DROP MLE ENV [ IF EXISTS ] [schema .] name
```

DROP MLE MODULE

```
 DROP MLE MODULE [ IF EXISTS ][schema .] module_name
```

DROP OPERATOR

```
DROP OPERATOR [ IF EXISTS ][ schema. ] operator [ FORCE ]
```

DROP OUTLINE

```
DROP OUTLINE outline
```

DROP PACKAGE

```
DROP PACKAGE [ IF EXISTS ] [ BODY ] [ schema. ] package
```

DROP PLUGGABLE DATABASE

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

DROP PMEM FILESTORE

```
DROP PMEM FILESTORE filestore_name
    [ FORCE INCLUDING CONTENTS
  | ( INCLUDING | EXCLUDING ) CONTENTS   ]
```

DROP PROCEDURE

```
DROP PROCEDURE [ IF EXISTS ][ schema. ] procedure
```

DROP PROFILE

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

DROP PROPERTY GRAPH

```
DROP PROPERTY GRAPH [ IF EXISTS ] [ schema . ] graph_name
```

DROP RESTORE POINT

```
DROP RESTORE POINT restore_point [ FOR PLUGGABLE DATABASE pdb_name ]
```

DROP ROLE

```
DROP ROLE role
```

DROP ROLLBACK SEGMENT

```
DROP ROLLBACK SEGMENT rollback_segment
```

DROP SEQUENCE

```
DROP SEQUENCE [ IF EXISTS ][ schema. ] sequence_name
```

DROP SYNONYM

```
DROP [PUBLIC] SYNONYM [ IF EXISTS ][ schema. ] synonym [FORCE]
```

DROP TABLE

```
DROP TABLE [ IF EXISTS ][ schema. ] table
  [ CASCADE CONSTRAINTS ] [ PURGE ]
```

DROP TABLESPACE

```
DROP TABLESPACE [ IF EXISTS ] tablespace
  [ { DROP | KEEP } QUOTA ]
  [ INCLUDING CONTENTS [ { AND | KEEP } DATAFILES ] [ CASCADE CONSTRAINTS ] ]
```

DROP TABLESPACE SET

```
DROP TABLESPACE SET tablespace_set
   [ { DROP | KEEP } QUOTA ]
   [ INCLUDING CONTENTS [ { AND | KEEP } DATAFILES ] [ CASCADE CONSTRAINTS ] ]
```

DROP TRIGGER

```
DROP TRIGGER [ IF EXISTS ][ schema. ] trigger
```

DROP TYPE

```
DROP TYPE [ IF EXISTS ][ schema. ] type_name [ FORCE | VALIDATE ]
```

DROP TYPE BODY

```
DROP TYPE BODY [ IF EXISTS ] [ schema. ] type_name
```

DROP USER

```
DROP USER [ IF EXISTS ]user [ CASCADE ]
```

DROP VIEW

```
DROP VIEW [ IF EXISTS ][ schema. ] view [ CASCADE CONSTRAINTS ]
```

EXPLAIN PLAN

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

FLASHBACK DATABASE

```
FLASHBACK [ STANDBY ] [ PLUGGABLE ] 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_schema_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 ]
         | values_clause
   ON ( condition )
   [ merge_update_clause ]
   [ merge_insert_clause ]
   [ error_logging_clause ]
   [ returning_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 [ { BY user [, user]... } | by_users_with_roles ]
    [ WHENEVER  [ NOT ]  SUCCESSFUL ] }
  |
  { CONTEXT NAMESPACE namespace ATTRIBUTES attribute [, attribute ]...
      [, CONTEXT NAMESPACE namespace ATTRIBUTES attribute [, attribute ]... ]...
    [ BY user [, user]... ]
  }
```

PURGE

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

RENAME

```
RENAME old_name TO new_name
```

REVOKE

```
REVOKE
  { { revoke_system_privileges | revoke_schema_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
   [ from_clause ]
   [ where_clause ]
   [ order_by_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 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 { window_name | (analytic_clause)}
```

ANY_VALUE

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

APPROX_COUNT

```
APPROX_COUNT ( ( expr [ , expr 'MAX_ERROR' ] ...) )
```

APPROX_COUNT_DISTINCT

```
APPROX_COUNT_DISTINCT(expr)
```

APPROX_COUNT_DISTINCT_AGG

```
APPROX_COUNT_DISTINCT_AGG(detail)
```

APPROX_COUNT_DISTINCT_DETAIL

```
APPROX_COUNT_DISTINCT_DETAIL(expr)
```

APPROX_MEDIAN

```
APPROX_MEDIAN( expr [ DETERMINISTIC ] [, { 'ERROR_RATE' | 'CONFIDENCE' } ] )
```

APPROX_PERCENTILE

```
APPROX_PERCENTILE( expr [ DETERMINISTIC ] [, { 'ERROR_RATE' | 'CONFIDENCE' } ] )
  WITHIN GROUP ( ORDER BY expr [ DESC | ASC ] )
```

APPROX_PERCENTILE_AGG

```
APPROX_PERCENTILE_AGG(expr)
```

APPROX_PERCENTILE_DETAIL

```
APPROX_PERCENTILE_DETAIL( expr [ DETERMINISTIC ] )
```

APPROX_RANK

```
APPROX_RANK ( expr [ PARTITION BY partition_by_clause ] [ ORDER BY order_by_clause
DESC] )
```

APPROX_SUM

```
APPROX_SUM ( expr [ , expr 'MAX_ERROR' ] ...)
```

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)
```

BIT_AND_AGG

```
BIT_AND_AGG ( [DISTINCT | ALL | UNIQUE] expr )
```

BITMAP_BIT_POSITION

```
BITMAP_BIT_POSITION  ( expr )
```

BITMAP_BUCKET_NUMBER

```
BITMAP_BUCKET_NUMBER ( expr )
```

BITMAP_CONSTRUCT_AGG

```
BITMAP_CONSTRUCT_AGG (  expr  )
```

BITMAP_COUNT

```
BITMAP_COUNT ( expr )
```

BITMAP_OR_AGG

```
BITMAP_OR_AGG ( expr )
```

BIT_OR_AGG

```
BIT_OR_AGG ( [DISTINCT | ALL | UNIQUE] expr )
```

BIT_XOR_AGG

```
BIT_XOR_AGG ( [DISTINCT | ALL | UNIQUE] expr )
```

BOOLEAN_AND_AGG

```
BOOLEAN_AND_AGG ( [ DISTINCT | ALL ] boolean_expr )
  [ OVER ( analytic_clause ) ]
```

BOOLEAN_OR_AGG

```
BOOLEAN_OR_AGG ( [ DISTINCT | ALL ] boolean_expr )
  [ OVER ( analytic_clause ) ]
```

CARDINALITY

```
CARDINALITY(nested_table)
```

CAST

```
CAST({ expr | MULTISET (subquery) } AS type_name
  [ DEFAULT return_value ON CONVERSION ERROR ]
  [, fmt [, 'nlsparam' ] ])
```

CEIL (datetime)

```
CEIL ( datetimes [ , fmt ] )
```

CEIL(interval)

```
CEIL ( interval [ , fmt ] )
```

CEIL(number)

```
CEIL(n)
```

CHARTOROWID

```
CHARTOROWID(char)
```

CHECKSUM

```
 CHECKSUM ( [ALL | DISTINCT | UNIQUE] expr )
```

CHR

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

CLUSTER_DETAILS (aggregate)

```
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 (aggregate)

```
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 (aggregate)

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

CLUSTER_ID (analytic)

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

CLUSTER_PROBABILITY (aggregate)

```
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 (aggregate)

```
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 ]...)
```

COLLATION

```
COLLATION(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_ID_TO_CON_NAME

```
CON_ID_TO_CON_NAME(container_guid)
```

CON_ID_TO_DBID

```
CON_ID_TO_DBID(container_guid)
```

CON_ID_TO_GUID

```
CON_ID_TO_GUID(container_guid)
```

CON_ID_TO_UID

```
CON_ID_TO_UID(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)
```

COSINE_DISTANCE

```
COSINE_DISTANCE ( expr1, expr2 )
```

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_MAT_PARTITION

```
DATAOBJ_TO_MAT_PARTITION( table, partition_id )
```

DATAOBJ_TO_PARTITION

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

DBTIMEZONE

```
DBTIMEZONE
```

DECODE

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

DECOMPOSE

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

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)
```

DOMAIN_CHECK

```
DOMAIN_CHECK( domain_name, ( expr )[ ,expr ]... )
```

DOMAIN_CHECK_TYPE

```
DOMAIN_CHECK_TYPE( domain_name, ( expr )[ ,expr ]... )
```

DOMAIN_DISPLAY

```
DOMAIN DISPLAY( domain_name, ( expr )[ ,expr ]... )
```

DOMAIN_NAME

```
DOMAIN_NAME( domain_name, ( expr )[ ,expr ]... )
```

DOMAIN_ORDER

```
DOMAIN_ORDER( domain_name, ( expr )[ ,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_COMPARE

```
FEATURE_COMPARE ( [ schema . ] model
  mining_attribute_clause AND mining_attribute_clause )
```

FEATURE_DETAILS (aggregate)

```
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 (aggregate)

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

FEATURE_ID (analytic)

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

FEATURE_SET (aggregate)

```
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 (aggregate)

```
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(datetime)

```
FLOOR ( datetimes [ , fmt ] )
```

FLOOR(interval)

```
FLOOR ( interval [ , fmt ] )
```

FLOOR(number)

```
FLOOR(n)
```

FROM_TZ

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

FROM_VECTOR

```
FROM_VECTOR
      ( expr [ RETURNING ( CLOB | VARCHAR2 [ ( size [BYTE | CHAR] ) ] ) ] )
```

GREATEST

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

GROUP_ID

```
GROUP_ID( )
```

GROUPING

```
GROUPING( { expr | c_alias } )
```

GROUPING_ID

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

HEXTORAW

```
HEXTORAW(char)
```

INITCAP

```
INITCAP(char)
```

INNER_PRODUCT

```
INNER_PRODUCT ( expr1 , expr2 )
```

INSTR

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

ITERATION_NUMBER

```
ITERATION_NUMBER
```

JSON_ARRAY

```
JSON_ARRAY
  ( JSON_ARRAY_content ) | JSON [ JSON_ARRAY_content ]
```

JSON_ARRAYAGG

```
JSON_ARRAYAGG
  ( expr [ FORMAT JSON ] [ order_by_clause ]
    [ JSON_on_null_clause ] [ JSON_returning_clause ]
    [ STRICT ] )
```

JSON_CONSTRUCTOR

```
JSON_CONSTRUCTOR ( expr )
```

JSON_DATAGUIDE

```
JSON_DATAGUIDE ( expr [ , format [ , flag ] ] )
```

JSON_MERGEPATCH

```
JSON_MERGEPATCH
   ( JSON_target_expr , JSON_patch_expr [ JSON_returning_clause ] [ PRETTY ] [ ASCII ]
     [ TRUNCATE ] [ JSON_on_error_clause ] )
```

JSON_OBJECT

```
JSON_OBJECT
    ( JSON_OBJECT_content ) | JSON { JSON_OBJECT_content }
```

JSON_OBJECTAGG

```
JSON_OBJECTAGG
  ( [ KEY ] key_expr VALUE val_expr [ FORMAT JSON ]
    [ JSON_on_null_clause ] [ JSON_returning_clause ]
    [ STRICT ] [ WITH UNIQUE KEYS ]  )
```

JSON_QUERY

```
JSON_QUERY
  ( expr [ FORMAT JSON ], JSON_basic_path_expression
    [ JSON_passing_clause ]
    [ JSON_query_returning_clause ]
    [ JSON_query_wrapper_clause ]
    [ JSON_query_quotes_clause ]
    [ JSON_query_on_error_clause ]
    [ JSON_query_on_empty_clause ]
    [ JSON_query_on_mismatch_clause ]
    [ TYPE ( { STRICT | LAX } ) ]
  )
```

JSON_SCALAR

```
JSON_SCALAR
( expr [ SQL | JSON ] [ NULL ON NULL ] [ NULL ON ERROR ] [ ERROR ON ERROR ]
  [ EMPTY STRING ON NULL ] )
```

JSON_SERIALIZE

```
JSON_SERIALIZE
( expr [ JSON_returning_clause ] [ PRETTY ] [ASCII ] [ ORDERED ][ TRUNCATE ]
   [ { NULL | ERROR | ( EMPTY { ARRAY | OBJECT } ) } ON ERROR ]
)
```

JSON_TABLE

```
JSON_TABLE
  ( expr [ FORMAT JSON ] [ , JSON_basic_path_expression ]
    [ JSON_table_on_error_clause ]
     [ TYPE ( { STRICT | LAX } ) ]
    JSON_columns_clause )
```

JSON_TRANSFORM

```
JSON_TRANSFORM ( input_expr , operation [ , operation ]...
    [ TYPE ( { STRICT | LAX } ) ]
    [ JSON_passing_clause ]
    [ JSON_TRANSFORM_returning_clause ] )
```

JSON Type Constructor

```
JSON ( expr )
```

JSON_VALUE

```
JSON_VALUE
  ( expr [ FORMAT JSON ] [ , JSON_basic_path_expression ] [ JSON_passing_clause ]
    [ JSON_value_returning_clause ] [ JSON_value_on_error_clause ]
    [ JSON_value_on_empty_clause ] [ JSON_value_on_mismatch_clause ]
    [ TYPE ( { STRICT | LAX } ) ]
  )
```

KURTOSIS_POP

```
KURTOSIS_POP ( [ {DISTINCT | ALL | UNIQUE} ]  expr )
```

KURTOSIS_SAMP

```
KURTOSIS_SAMP ( [ {DISTINCT | ALL | UNIQUE} ] x expr )
```

L1_DISTANCE

```
L1_DISTANCE ( expr1 , expr2 )
```

L2_DISTANCE

```
L2_DISTANCE ( expr1 , expr2 )
```

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( [ALL | DISTINCT ] measure_expr
           [, 'delimiter'] [listagg_overflow_clause] )
           [ 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_COLLATION_ID

```
NLS_COLLATION_ID(expr)
```

NLS_COLLATION_NAME

```
NLS_COLLATION_NAME(expr [, flag ])
```

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_DM_PARTITION_NAME

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

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 (aggregate)

```
PREDICTION ( [ grouping_hint ] [ 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 (aggregate)

```
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 (aggregate)

```
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 (aggregate)

```
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 (aggregate)

```
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 (datetime)

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

ROUND(interval)

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

ROUND (number)

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

ROUND_TIES_TO_EVEN (number)

```
ROUND_TIES_TO_EVEN ( 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)
```

SKEWNESS_POP

```
SKEWNESS_POP [ DISTINCT | ALL | UNIQUE ] ( expr )
```

SKEWNESS_SAMP

```
 SKEWNESS_SAMP [DISTINCT | ALL | UNIQUE] ( expr )
```

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)
```

TIME_BUCKET (datetime)

```
TIME_BUCKET ( datetime , stride, origin, [ START | END ] [ timebucket_optional_clause ])
```

TO_APPROX_COUNT_DISTINCT

```
TO_APPROX_COUNT_DISTINCT(detail)
```

TO_APPROX_PERCENTILE

```
TO_APPROX_PERCENTILE(detail, expr, 'datatype'
  [, { 'DESC' | 'ASC' | 'ERROR_RATE' | 'CONFIDENCE' } ])
```

TO_BINARY_DOUBLE

```
TO_BINARY_DOUBLE(expr [ DEFAULT return_value ON CONVERSION ERROR ]
  [, fmt [, 'nlsparam' ] ])
```

TO_BINARY_FLOAT

```
TO_BINARY_FLOAT(expr [ DEFAULT return_value ON CONVERSION ERROR ]
  [, fmt [, 'nlsparam' ] ])
```

TO_BLOB (bfile)

```
TO_BLOB( bfile [, mime_type] )
```

TO_BLOB (raw)

```
TO_BLOB( raw_value )
```

TO_BOOLEAN

```
TO_BOOLEAN( expr )
```

TO_CHAR (bfile|blob)

```
TO_CHAR( { bfile | blob } [, csid] )
```

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 (bfile|blob)

```
TO_CLOB( { bfile | blob } [, csid] [, mime_type] )
```

TO_CLOB (character)

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

TO_DATE

```
TO_DATE(char [ DEFAULT return_value ON CONVERSION ERROR ]
  [, fmt [, 'nlsparam' ] ])
```

TO_DSINTERVAL

```
TO_DSINTERVAL ( ' { sql_format | ds_iso_format } '
  [ DEFAULT return_value ON CONVERSION ERROR ] )
```

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 [ DEFAULT return_value ON CONVERSION ERROR ]
  [, fmt [, 'nlsparam' ] ])
```

TO_SINGLE_BYTE

```
TO_SINGLE_BYTE(char)
```

TO_TIMESTAMP

```
TO_TIMESTAMP(char [ DEFAULT return_value ON CONVERSION ERROR ]
  [, fmt [, 'nlsparam' ] ])
```

TO_TIMESTAMP_TZ

```
TO_TIMESTAMP_TZ(char [ DEFAULT return_value ON CONVERSION ERROR ]
  [, fmt [, 'nlsparam' ] ])
```

TO_UTC_TIMESTAMP_TZ

```
TO_UTC_TIMESTAMP_TZ ( varchar )
```

TO_VECTOR

```
TO_VECTOR ( expr [ , number_of_dimensions [ , format ] ] )
```

TO_YMINTERVAL

```
TO_YMINTERVAL
  ( '  { [+|-] years - months
       | ym_iso_format
       } '
    [ DEFAULT return_value ON CONVERSION ERROR ]
  )
```

TRANSLATE

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

TRANSLATE ... USING

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

TREAT

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

TRIM

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

TRUNC (datetime)

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

TRUNC(interval)

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

TRUNC (number)

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

TZ_OFFSET

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

UID

```
UID
```

UNISTR

```
UNISTR( string )
```

UPPER

```
UPPER(char)
```

USER

```
USER
```

user-defined function

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

USERENV

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

VALIDATE_CONVERSION

```
VALIDATE_CONVERSION(expr AS type_name
  [, fmt [, 'nlsparam' ] ])
```

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) ]
```

VECTOR

```
VECTOR ( expr [ , number_of_dimensions [ , format ] ] )
```

VECTOR_CHUNKS

```
VECTOR_CHUNKS ( chunks_table_arguments )
```

VECTOR_DISTANCE

```
VECTOR_DISTANCE ( expr1 , expr2 [ , metric ] )
```

VECTOR_DIMS

```
VECTOR_DIMS ( expr )
```

VECTOR_DIMENSION_COUNT

```
VECTOR_DIMENSION_COUNT ( expr )
```

VECTOR_DIMENSION_FORMAT

```
VECTOR_DIMENSION_FORMAT ( expr )
```

VECTOR_EMBEDDING

```
 VECTOR_EMBEDDING ( [ schema.]model_name USING mining_attribute_clause )
```

VECTOR_NORM

```
VECTOR_NORM ( expr )
```

VECTOR_SERIALIZE

```
VECTOR_SERIALIZE ( expr [ RETURNING ( CLOB | VARCHAR2 [ ( size [BYTE | CHAR] ) ] ) ] )
```

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]
 )
```

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 Subclauses for the syntax of the subclauses.

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

Calculated Measure Expressions

```
{    av_meas_expression
   | av_simple_expression
   | single_row_function_expression
   | case_expression
   | compound_expression
   | datetime_expression
   | interval_expression
}
```

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
| expr COLLATE collation_name
```

```
}

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 [ array_step ]... ]...
```

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

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_modifier_list ] JSON IS_JSON_args
```

IS JSON Condition

```
 [ FORMAT JSON ][ ( {STRICT | LAX} ) ] [ { WITH | WITHOUT} UNIQUE KEYS ]
             |  VALIDATE [CAST] [USING] schema
```

IS OF *type* condition

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

IS PRESENT condition

```
cell_reference IS PRESENT
```

JSON_EQUAL condition

```
JSON_EQUAL ( (expr), (expr) )
```

JSON_EXISTS condition

```
JSON_EXISTS( expr [ FORMAT JSON ], JSON_basic_path_expression
  [ JSON_passing_clause]
  [ JSON_exists_on_error_clause ] [ TYPE ( { STRICT | LAX } ) ]
  [ JSON_exists_on_empty_clause ] )
```

JSON_TEXTCONTAINS condition

```
JSON_TEXTCONTAINS( column, JSON_basic_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:

- SQL Statements
- SQL Functions
- SQL Expressions
- SQL Conditions

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

abbreviated_edge_pattern

```
->
  | <-
  | -
```

action_audit_clause

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

activate_standby_db_clause

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

accuracy_clause

```
[ WITH ] [ TARGET ]
  ACCURACY { accuracy [PERCENT]
             |  PARAMETERS  { ( EFSEARCH efs [, NEIGHBOR PARTITION PROBES nprobes] )
                  | ( NEIGHBOR PARTITION PROBES nprobes [, EFSEARCH efs] )
             }
           }
```

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_definition | virtual_column_definition } ]...
         [ domain_definition [, domain_definition ]...
        )
   [ column_properties ]
   [ ( out_of_line_part_storage [, out_of_line_part_storage]... ) ]
```

add_disk_clause

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

add_external_partition_attrs

```
ADD EXTERNAL PARTITION ATTRIBUTES external_table_clause
 [ REJECT LIMIT ]
```

add_filegroup_clause

```
ADD FILEGROUP filegroup_name
  { DATABASE database_name
  | CLUSTER cluster_name
  | VOLUME asm_volume
  | TEMPLATE               }
  [ FROM TEMPLATE <template_name> ]
  }
[ SET '[ file_type. ] property_name' = 'property_value' ]
```

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 ]
[ read_only_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 ]
[ external_part_subpart_data_props ]
[ ( { 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_meas_clause

```
ADD MEASURES  ( (cube_meas)...)
```

add_mv_log_column_clause

```
ADD (column)
```

add_overflow_clause

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

add_partitionset

```
ADD { range_partitionset_clause | list_partitionset_clause }
```

add_period_clause

```
ADD ( period_definition )
```

add_range_partition_clause

```
range_values_clause
[ table_partition_description ]
[ external_part_subpart_data_props ]
[ ( { 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' ]
  [ FORCE KEYSTORE ]
  IDENTIFIED BY { EXTERNAL STORE | keystore_password }
  [ WITH BACKUP [ USING 'backup_identifier' ] ]
```

add_update_secret_seps

```
{ ADD | UPDATE } SECRET 'secret' FOR CLIENT 'client_identifier'
  [ USING TAG 'tag' ]
  TO  [ LOCAL ] AUTO_LOGIN KEYSTORE 'directory'
```

add_volume_clause

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

advanced_index_compression

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

affinity_clauses

```
{ ENABLE AFFINITY [ schema.]table [SERVICE service_name ]
|
DISABLE AFFINITY [ schema.]table
}
```

alias

```
name
```

alias_file_name

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

all_clause

```
ALL  MEMBER { NAME expression [ MEMBER CAPTION expression ]
            | CAPTION expression [ MEMBER DESCRIPTION expression ]
            | DESCRIPTION expression
            }
```

allocate_extent_clause

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

allow_disallow_clustering

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

alter_add_cache_clause

```
ADD CACHE
  MEASURE GROUP [ ALL | ( meas_name )... ]
  LEVELS ( [ [ dim_alias "." ] hier_alias "." ] level )...
```

alter_automatic_partitioning

```
{ SET PARTITIONING { AUTOMATIC | MANUAL }
| SET STORE IN ( tablespace [, tablespace ]... )
}
```

alter_datafile_clause

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

alter_drop_cache_clause

```
DROP CACHE
  MEASURE GROUP [ ALL | ( meas_name )... ]
  LEVELS ( [ [ dim_alias "." ] hier_alias "." ] level )...
```

alter_external_table

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

```
  | parallel_clause
  | external_table_data_props
  | 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
  [ FORCE KEYSTORE ]
  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 [ SERVICE = service_name ]
    | ROW ARCHIVAL VISIBILITY = { ACTIVE | ALL }
    | DEFAULT_COLLATION = { collation_name | NONE }
    }
```

alter_system_reset_clause

```
parameter_name
   [ { SCOPE = { MEMORY | SPFILE | BOTH }
     | 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_automatic_partitioning
| alter_interval_partitioning
| set_subpartition_template
| modify_table_partition
| modify_table_subpartition
| move_table_partition
| move_table_subpartition
| add_external_partition_attrs
| add_table_partition
| coalesce_table_partition
| drop_external_partition_attrs
| 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_partitionset

```
[ {    add_partitionset
     |  modify_partitionset
     |  move_partitionset
     | split_partitionset
   }
]
```

alter_table_properties

```
{ { { physical_attributes_clause
    | logging_clause
    | table_compression
    | inmemory_table_clause
    | ilm_clause
    | supplemental_table_logging
    | allocate_extent_clause
    | deallocate_unused_clause
    | { CACHE | NOCACHE }
    | result_cache_clause
    | upgrade_table_clause
    | records_per_block_clause
    | parallel_clause
    | row_movement_clause
    | logical_replication_clause
    | flashback_archive_clause
    }...
  | RENAME TO new_table_name
  } [ alter_iot_clauses ] [ alter_XMLSchema_clause ]
| { shrink_clause
  | READ { ONLY | WRITE }
  | REKEY encryption_spec
  | DEFAULT COLLATION collation_name
  | [NO] ROW ARCHIVAL
  | ADD attribute_clustering_clause
  | MODIFY CLUSTERING [ clustering_when ] [ zonemap_clause ]
  | DROP CLUSTERING
  | [ [ NOT ] FOR STAGING ]
  | annotations_clause
  }
}
```

alter_tablespace_attrs

```
{ default_tablespace_params
| 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_tablespace_encryption
}
```

alter_tablespace_encryption

```
ENCRYPTION
  { { ONLINE { { [ tablespace_encryption_spec ] { ENCRYPT | REKEY } }
               | DECRYPT } [ ts_file_name_convert ] }
  | { FINISH { ENCRYPT | REKEY | DECRYPT } [ ts_file_name_convert ] }
  | { OFFLINE [ tablespace_encryption_spec ] { ENCRYPT | DECRYPT }
  }
```

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 }
}...
```

alternate_key_clause

```
ALTERNATE KEY { [ ( ] attribute [ ) ]
                |
                ( attribute [, attribute ]... )
              }
```

analytic_clause

```
[ { query_partition_clause | window_name } ] [ order_by_clause [ windowing_clause ] ]
```

annotations_clause

```
ANNOTATIONS ( annotations_list )
```

annotations_list

```
 [ ADD [ IF NOT EXISTS | OR REPLACE ] | DROP [ IF EXISTS ] | REPLACE ] annotation
 [ , [ ADD [ IF NOT EXISTS | OR REPLACE ] | DROP [ IF EXISTS ] | REPLACE ] annotation ]...
```

annotation

```
annotation_name [ annotation_value ]
```

annotation_name

```
identifier
```

annotation_value

```
character_string_literal
```

append_op

```
APPEND pathExpr = rhsExpr
   [ { IGNORE | ERROR | CREATE | NULL } ON MISSING ]
   [ { IGNORE | ERROR | REPLACE | CREATE } ON MISMATCH ]
   [ { IGNORE | ERROR | NULL } ON NULL ]
   [ { IGNORE | ERROR } ON EMPTY ]
```

application_clauses

```
APPLICATION
{ { app_name
    { BEGIN INSTALL 'app_version' [ COMMENT 'comment' ]
    | END INSTALL [ 'app_version' ]
    | BEGIN PATCH number [ MINIMUM VERSION 'app_version' ] [ COMMENT 'comment' ]
    | END PATCH [ number ]
    | BEGIN UPGRADE ['start_app_version'] TO 'end_app_version' [ COMMENT 'comment' ]
    | END UPGRADE [ TO 'end_app_version' ]
    | BEGIN UNINSTALL
    | END UNINSTALL
    | SET PATCH number
    | SET VERSION 'app_version'
    | SET COMPATIBILITY VERSION { 'app_version' | CURRENT }
    | SYNC TO  { 'app_version' | PATCH patch_number }
    | [( app_name )...] SYNC
    }
  }
  |
  { ALL [ EXCEPT (app_name)... ] SYNC }
}
```

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' ]
   }
```

argument

```
name : value
```

arguments

```
 ( argument ) ...
```

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
}
```

attr_dim_attributes_clause

```
[ alias. ] column [ [ AS ] attribute_name ] [ classification_clause ]...
```

attr_dim_level_clause

```
LEVEL level [ { NOT NULL  |  SKIP WHEN NULL } ]
  [ classification_clause [ classification_clause ]...
  [ LEVEL TYPE
      {   STANDARD
        | YEARS
        | HALF_YEARS
        | QUARTERS
        | MONTHS
        | WEEKS
        | DAYS
        | HOURS
        | MINUTES
        | SECONDS
      }
  ]
  key_clause [ alternate_key_clause ]
  [ MEMBER NAME expression ]
  [ MEMBER CAPTION expression ]
  [ MEMBER DESCRIPTION expression ]
  [ ORDER BY  [ MIN | MAX ] dim_order_clause
                   [, [ MIN | MAX ] dim_order_clause ]... ]
  ]
  [ DETERMINES ( attribute [, attribute]... ) ]
```

attr_dim_using_clause

```
USING (source_clause)... [ (join_path_clause)...
```

attribute_clause

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

attribute_clustering_clause

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

attributes_clause

```
ATTRIBUTES ( attr_dim_attribute_clause [, attr_dim_attribute_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 [, sql_operation ]
| 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 ]
   }
```

av_meas_expression

```
{  lead_lag_expression
 | window_expression
 | share_of_expression
 | qdr_expression
}
```

av_measure

```
meas_name  [{ base_measure_clause | calc_measure_clause }]
  [ classification_clause ]...
```

av_simple_expression

```
{ string | number | NULL | measure_ref }
```

av_window_clause

```
HIERARCHY hierarchy_ref
{ IN member_set
 | BETWEEN { preceding_boundary | following_boundary }
 [ WITHIN {   LEVEL | PARENT | ANCESTOR AT LEVEL level_ref } ]
}
```

backup_keystore

```
BACKUP KEYSTORE [ USING 'backup_identifier' ]
  [ FORCE KEYSTORE ]
  IDENTIFIED BY { EXTERNAL STORE | keystore_password }
  [ TO 'keystore_location' ]
```

base_meas_clause

```
[ FACT [ ( ( expression )... ) ]  ] [ meas_aggregate_clause ]
```

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
```

blockchain_table_clauses

```
blockchain_drop_table_clause
     blockchain_row_retention_clause
     blockchain_hash_clause
     [ blockchain_row_version_user_chain_clause ]
     [ blockchain_system_chains_clause ]
       blockchain_data_format_clause
```

blockchain_drop_table_clause

```
 NO DROP [ UNTIL integer DAYS IDLE ]
```

blockchain_row_retention_clause

```
NO DELETE { [ LOCKED ] | (UNTIL integer DAYS AFTER INSERT [ LOCKED ]) }
```

blockchain_hash_clause

```
HASHING USING SHA2_512
```

blockchain_row_version_user_chain_clause

```
[ WITH { (USER CHAIN) | ( ROW VERSION  [ AND USER CHAIN ] ) } row_version_name
    ( ( column )[, column ...] ) ]
```

blockchain_system_chains_clause

```
CONFIGURE integer SYSTEM CHAINS PER INSTANCE
```

blockchain_data_format_clause

```
VERSION ( v1 | v2 )
```

boolean_expression

```
condition
```

boolean_test_condition

```
boolean_expression IS [ NOT ] { TRUE| FALSE| NULL }
```

build_clause

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

by_users_with_roles

```
BY USERS WITH GRANTED ROLES role [, role]...
```

cache_clause

```
CACHE cache_specification [, cache_specification]...
```

cache_specification

```
MEASURE GROUP
 {    ALL
   | ( measure_name [, measure_name ]... ) [ levels_clause ]...
 }
```

calc_meas_order_by_clause

```
calc_meas_expression [ { ASC | DESC } ]  [ NULLS { FIRST | LAST } ]
```

calc_meas_clause

```
AS ( expression )
```

cancel_sql_clause

```
CANCEL SQL ' session_id , serial_number [ , @ instance_id ] [ , sql_id ] '
```

case_op

```
CASE ( WHEN path_expr THEN operation [ , operation ]... )...
           [ ELSE ( operation [ , operation ]... )] END
```

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 ]
```

chunking_mode

```
WORDS
         | CHARS
         | CHARACTERS
         | VOCABULARY vocabulary_name
```

classification_clause

```
[ CAPTION caption ]
[ DESCRIPTION description ]
[ CLASSIFICATION classification_name
  [ VALUE classification_value ]
  [ LANGUAGE language ]
]...
```

clause_options

```
OPTION
{ { = ( 'clause_option' | 'clause_option_pattern'
        [, 'clause_option' | 'clause_option_pattern' ]... ) }
| { = ( 'clause_option' ) option_values }
| { ALL [ EXCEPT = ( 'clause_option' | 'clause_option_pattern'
                    [, 'clause_option' | 'clause_option_pattern' ]... ) ] }
}
```

clear_free_space_clause

```
CLEAR FREE SPACE
```

close_keystore

```
SET KEYSTORE CLOSE
  [ IDENTIFIED BY { EXTERNAL STORE | 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 ...
|  rename_LOB_storage_clause
```

```
|  alter_varray_col_properties ...
}
```

column_definition

```
column [ datatype_domain ]
  [ [ COLLATE column_collation_name ] | RESERVABLE ]
  [ SORT ] [ VISIBLE | INVISIBLE ]
   [ DEFAULT [ ON NULL [ FOR ( INSERT { ONLY | AND UPDATE } ) ] ]
  expr
  | identity_clause ]
  [ ENCRYPT encryption_spec ]
  [ ( inline_constraint )... | inline_ref_constraint ]
  [ annotations_clause ]
```

column_name_list

```
( column_name )...
```

column_or_expression

```
( ( column_name [ AS property_name ] ) | ( value_expression   AS property_name ) )
```

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
| json_storage_clause
}...
```

column_tags_clause

```
WITH { { CHECK | NOCHECK } [ ETAG ]
  | UPDATE
  | NOUPDATE }...
```

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 ]...
  | PROTOCOL { HTTP | FTP | AUTHENTICATION }
```

composite_directory_based_partitions

```
PARTITION BY DIRECTORY ( (column_name)[, column_name ])
  { subpartition_by_range
  | subpartition_by_list
```

```
  | subpartition_by_hash
  }
  ( ( directory_partition_desc )[, directory_partition_desc ] )
```

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 [, column]... )
[ AUTOMATIC [ STORE IN ( tablespace [, tablespace ]... ) ] ]
  { 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]... )
[ DIRECTORY TABLESPACE tablespace_name ]
```

condition_clause

```
 { tracking_statistics_clause |  ( ON PLSQL_function_name ) }
```

conditional_insert_clause

```
[ ALL | FIRST ]
WHEN condition
THEN insert_into_clause
  [ insert_values_clause ]
  [ error_logging_clause ]
  [ insert_into_clause [ insert_values_clause ] [ error_logging_clause ] ]...
[ WHEN condition
  THEN insert_into_clause
    [ insert_values_clause ]
    [ error_logging_clause ]
    [ insert_into_clause [ insert_values_clause ] [ error_logging_clause ] ]...
]...
[ ELSE insert_into_clause
  [ insert_values_clause ]
  [ error_logging_clause ]
   [ insert_into_clause [ insert_values_clause ] [ error_logging_clause ] ]...
]
```

consistent_hash_partitions

```
PARTITION BY CONSISTENT HASH (column [, column ]...)
  [ PARTITIONS AUTO ] TABLESPACE SET tablespace_set
```

consistent_hash_with_subpartitions

```
PARTITION BY CONSISTENT HASH (column [, column ]...)
  { subpartition_by_range
  | subpartition_by_list
  | subpartition_by_hash
  }
  [ PARTITIONS AUTO ]
```

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 ] [ precheck_state ]
| 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 ]
```

container_map_clause

```
CONTAINER_MAP UPDATE { add_table_partition | split_table_partition }
```

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
```

convert_redundancy_clause

```
CONVERT TO FLEX REDUNDANCY
```

copy_op

```
COPY pathExpr = rhsExpr
    [ { IGNORE | ERROR | CREATE | NULL } ON MISSING ]
    [ { IGNORE | ERROR | NULL } ON NULL ]
    [ { IGNORE | ERROR } ON EMPTY ]
```

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_single_column_domain

```
CREATE DOMAIN [IF NOT EXISTS ][ schema .] domain_name AS datatype
[column_properties_clause]
 | [ DISPLAY display_expression ]
 | [ ORDER order_expression ]
 | [ annotations_clause ]
```

create_multi_column_domain

```
CREATE DOMAIN [ IF NOT EXISTS ][ schema .] domain_name AS
( domain_column AS datatype [column_properties_clause] [, column_properties_clause] )
 | [DISPLAY display_expression ]
 | [ORDER order_expression ]
 | [annotations_clause ]
```

create_flexible_domain

```
CREATE FLEXIBLE DOMAIN [IF NOT EXISTS ][ schema .]domain_name
    ( domain_column [ , domain_column... ] )
    CHOOSE DOMAIN USING ( domain_discriminant_column (datatype)[ , datatype...] )
FROM
  ( DECODE (expr , search_expr , result_expr [, search_expr ,( result_expr )[ ,
result_expr]...]... )
   | CASE ( expr (WHEN comparison_expr THEN return_expr )...
   | ( WHEN condition THEN return_expr)... )
   [ ELSE else_expr ]
   END )
```

create_file_dest_clause

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

create_key

```
CREATE [ ENCRYPTION ] KEY { mkid:mk | mk }
  [ USING TAG 'tag' ]
  [ USING ALGORITHM 'encrypt_algorithm' ]
  [ FORCE KEYSTORE ]
  IDENTIFIED BY { EXTERNAL STORE | 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
    | ON STATEMENT
    }
  | { 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 } }
|
  { AS PROXY FROM src_pdb_name @ dblink }
}
  [ parallel_pdb_creation_clause ]
  [ default_tablespaces ]
  [ pdb_storage_clause ]
  [ file_name_convert ]
```

```
  [ service_name_convert ]
  [ path_prefix_clause ]
  [ tempfile_reuse_clause ]
  [ SNAPSHOT COPY ]
  [ user_tablespaces_clause ]
  [ standbys_clause ]
  [ logging_clause ]
  [ create_file_dest_clause ]
  [ keystore_clause ]
  [ pdb_refresh_mode_clause ]
  [ RELOCATE [ KEEP SOURCE ] [ AVAILABILITY {MAX | NORMAL} ]
      pdb_refresh_mode_clause ]
  [ NO DATA ]
  [ HOST = 'hostname' ]
  [ PORT = number ]
```

create_pdb_from_mirror_copy

```
new_pdb_name FROM base_pdb_name @dblinkname
USING MIRROR COPY mirror_name
```

create_pdb_from_seed

```
ADMIN USER admin_user_name IDENTIFIED BY password
  [ pdb_dba_roles ]
  [ parallel_pdb_creation_clause ]
  [ default_tablespace ]
  [ pdb_storage_clause ]
  [ file_name_convert ]
  [ service_name_convert ]
  [ path_prefix_clause ]
  [ tempfile_reuse_clause ]
  [ user_tablespaces_clause ]
  [ standbys_clause ]
  [ logging_clause ]
  [ create_file_dest_clause ]
  [ HOST = 'hostname' ]
  [ PORT = number ]
```

create_pdb_from_xml

```
[ AS CLONE ] USING filename
  [ source_file_name_convert | source_file_directory ]
  [ { [ COPY | MOVE ] file_name_convert } | NOCOPY ]
  [ service_name_convert ]
  [ default_tablespace ]
  [ pdb_storage_clause ]
  [ path_prefix_clause ]
  [ tempfile_reuse_clause ]
  [ user_tablespaces_clause ]
  [ standbys_clause ]
  [ logging_clause ]
  [ create_file_dest_clause ]
  [ HOST = 'hostname' ]
  [ PORT = number ]
```

create_zonemap_as_subquery

```
CREATE MATERIALIZED ZONEMAP [ IF NOT EXISTS ]
  [ schema. ] zonemap_name
  [ zonemap_attributes ]
  [ zonemap_refresh_clause ]
  [ { ENABLE | DISABLE } PRUNING ]
  AS query_block
```

create_zonemap_on_table

```
CREATE MATERIALIZED ZONEMAP [ IF NOT EXISTS ]
  [ 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 }
```

cube_meas

```
 meas_name( base_meas_clause | calc_meas_clause )
```

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
| SET STANDBY NOLOGGING FOR {DATA AVAILABILITY | LOAD PERFORMANCE}
}
```

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 ]
```

datatype_domain

```
( datatype [ DOMAIN [domain_owner.] domain_name ]
  | [ DOMAIN ] [domain_owner.] domain_name )
```

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_aggregate_clause

```
DEFAULT AGGREGATE BY aggr_function
```

default_clause

```
DEFAULT [ ON NULL [FOR ( INSERT { ONLY | AND UPDATE } ) ] ] default_expression
```

default_cost_clause

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

default_index_compression

```
INDEX { COMPRESS ADVANCED { LOW | HIGH }
      | NOCOMPRESS
      }
```

default_measure_clause

```
DEFAULT MEASURE measure
```

default_selectivity_clause

```
DEFAULT SELECTIVITY default_selectivity
```

default_settings_clauses

```
{ DEFAULT EDITION = edition_name
| SET DEFAULT { BIGFILE | SMALLFILE } TABLESPACE
| DEFAULT TABLESPACE tablespace
| DEFAULT [ LOCAL ] 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
| CONTAINERS DEFAULT TARGET = { (container_name) | NONE }
```

```
| flashback_mode_clause
| undo_mode_clause
| set_time_zone_clause
}
```

default_table_compression

```
TABLE { COMPRESS FOR OLTP
      | COMPRESS FOR QUERY { LOW | HIGH }
      | COMPRESS FOR ARCHIVE { LOW | HIGH }
      | NOCOMPRESS
      }
```

default_tablespace

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

default_tablespace_params

```
DEFAULT [ default_table_compression ] [ default_index_compression ]
        [ inmemory_clause ] [ ilm_clause ] [ storage_clause ]
```

default_temp_tablespace

```
[ BIGFILE | SMALLFILE ] DEFAULT
{ { TEMPORARY TABLESPACE }
| { LOCAL TEMPORARY TABLESPACE FOR { ALL | LEAF } }
} tablespace
[ TEMPFILE file_specification [, file_specification ]...]
[ extent_management_clause ]
```

deferred_segment_creation

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

delete_secret

```
DELETE SECRET FOR CLIENT 'client_identifier'
  [ FORCE KEYSTORE ]
  IDENTIFIED BY { EXTERNAL STORE | keystore_password }
  [ WITH BACKUP [ USING 'backup_identifier' ] ]
```

delete_secret_seps

```
DELETE SECRET 'secret' FOR CLIENT 'client_identifier'
  FROM [ LOCAL ] AUTO_LOGIN KEYSTORE 'directory'
```

dependent_tables_clause

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

destination_predicate

```
vertex_reference IS [NOT] DESTINATION OF edge_reference
```

digit

```
{ 0 | ... | 9 }
```

dim_by_clause

```
DIMENSION BY ( dim_key [, dim_key ]... )
```

dim_key

```
dim_ref
  [classification_clause]...
  KEY
    {[() [alias.] fact_column  ())]
     |
     (  [alias.] fact_column [, [alias.] fact_column]... )
    }
  REFERENCES
     {[()  attribute  ())]
      |
      ( attribute [, attribute]... )
     }
  HIERARCHIES ( hier_ref [, hier_ref]... )
```

dim_order_clause

```
attribute [ ASC | DESC ] [ NULLS { FIRST | LAST } ]
```

dim_ref

```
[ schema. ] attr_dim_name [ [AS] dim__alias ]
```

dimension_join_clause

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

directive

```
@ name [ arguments ]
```

directives

```
directive ...
```

directory_based_partitions

```
PARTITION BY DIRECTORY ( (column_name)[ , column_name]...)
( ( PARTITION [ partition ] table_partition_description )[ , PARTITION [ partition ]
    table_partition_description ] )
    [ DIRECTORY TABLESPACE tablespace_name ]
```

disk_offline_clause

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

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 ]
```

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_external_partition_attrs

```
DROP EXTERNAL PARTITION ATTRIBUTES
```

drop_filegroup_clause

```
DROP FILEGROUP filegroup_name [ CASCADE ]
   [ FOR [ PLUGGABLE DATABASE pdb_name ] DATABASE db_unique_name ]
```

drop_index_partition

```
DROP PARTITION partition_name
```

drop_logfile_clauses

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

drop_mirror_copy

```
  DROP MIRROR COPY mirror_name
```

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 ] ]
```

duality_view_replication_clause

```
{ DISABLE | ENABLE } LOGICAL REPLICATION
```

duality_view_subquery

```
SELECT object_gen_clause FROM child_table [ child_table_alias ]
     [table_tags_clause] WHERE join_condition
```

dynamic_base_profile

```
INCLUDING base_profile
```

edge_pattern

```
{ full_edge_pattern | abbreviated_edge_pattern  }
```

edge_tables_clause

```
EDGE TABLES ( ( edge_table_definition)... )
```

edge_tables_definition

```
graph_element_name_and_key SOURCE vertex_table_reference
           DESTINATION vertex_table_reference [ graph_table_label_and_properties ]
```

element_pattern

```
{ vertex_pattern | edge_pattern }
```

element_pattern_where_clause

```
WHERE search_condition
```

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 ]
  ]
  [ undo_mode_clause ]
```

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 ]
```

entry

```
( regular_entry [ format_clause ] ) | wildcard
```

enum_alias_list

```
 = name enum_alias_list
```

enum_item_list

```
 name [ enum_alias_list ] [ = value ]
```

enum_list

```
enum_item_list [ , enum_item_list ...]
```

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'
  [ FORCE KEYSTORE ]
  IDENTIFIED BY keystore_password
  [ WITH IDENTIFIER IN { 'key_id' [, 'key_id' ]... | ( subquery ) } ]
```

expr

```
{ simple_expression
| compound_expression
| calc_meas_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
| boolean_expression
}
```

expression_list

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

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_part_subpart_data_props

```
[ DEFAULT DIRECTORY directory ]
[ LOCATION
   ([ directory: ] 'location_specifier'
      [, [ directory: ] 'location_specifier' ]...
   )
]
```

external_table_clause

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

external_table_data_props

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

fact_columns_clause

```
FACT COLUMNS ( fact_column [ ( [ AS ] fact_alias )... ] )
```

failover_clause

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

fetch_clause

```
FETCH [ EXACT | APPROX | APPROXIMATE ] { FIRST | NEXT }
```

field

```
[ alias : ] query_field_name [ directives ] [ selection_set ]
```

field_name

```
{ scalar_field_name | object_field_name }
```

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
}
```

filegroup_clauses

```
{ add_filegroup_clause
| modify_filegroup_clause
| move_to_filegroup_clause
| drop_filegroup_clause
}
```

filter_clause

```
 hier_ids TO predicate
```

filter_clauses

```
 FILTER FACT ( filter_clause ...)
```

filter_condition

```
INCLUDING ROWS where_clause
```

fixed_quantifier

```
( unsigned_integer )
```

flashback_archive_clause

```
[ BLOCKCHAIN ] 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
}
```

flex_clause

```
column_name AS FLEX [ COLUMN ]
```

flush_clause

```
FLUSH {  SQL_MONITOR | QUERY_HISTORY | SHARED_POOL | GLOBAL CONTEXT
                           | { BUFFER_CACHE  | FLASH_CACHE } [ LOCAL | GLOBAL ] }
            | REDO TO target_db_name [ [ NO ] CONFIRM APPLY ]
            | PASSWORDFILE_METADATA_CACHE
             }
```

following_boundary

```
{ CURRENT MEMBER | offset_expr FOLLOWING }
AND
{ offset_expr FOLLOWING | UNBOUNDED FOLLOWING }
```

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
    }
  ]
```

format_clause

```
FORMAT JSON
```

from_clause

```
FROM ( { table_reference | join_clause | { { join_clause ) | inline_analytic_view }})
  [ , { table_reference | join_clause | { { join_clause ) | inline_analytic_view }} ...]
  [ { table_reference | join_clause | { join_clause ) | inline_analytic_view } }
  [ , { join_clause ) | inline_analytic_view } } ...]
```

from_using_clause

```
{ FROM | USING }
       { table_reference | join_clause | { join_clause  | inline_analytic_view } }
       [ , { table_reference | join_clause | { join_clause | inline_analytic_view } }... ]
```

full_database_recovery

```
[ STANDBY ] DATABASE
[ { UNTIL { CANCEL
          | TIME date
          | CHANGE integer
          | CONSISTENT
          }
  | USING BACKUP CONTROLFILE
  | SNAPSHOT TIME date
  }...
]
```

full_edge_any_direction

```
 -[ optional_element_pattern_filter ]-
```

full_edge_pattern

```
 full_edge_pointing_right
| full_edge_pointing_left
| full_edge_any_direction
```

full_edge_pointing_left

```
 <- [ optional_element_pattern_filter ] -
```

full_edge_pointing_right

```
 -[ optional_element_pattern_filter ]->
```

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_quantifier

```
( [ lower_bound ] , upper_bound )
```

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 ]...
```

graph_element_key

```
KEY ( column_name_list )
```

graph_element_name_and_key

```
graph_element_object_name [ ( AS (graph_element_name) ) ] [ graph_element_key ]
```

graph_element_object_name

```
schema.( materialized_view_name | table_name | synonym )
```

graph_name

```
schema_qualified_name
```

graph_options

```
OPTIONS ( ( ( ENFORCED | TRUSTED ) MODE
              | ( ( ALLOW | DISALLOW ) MIXED PROPERTY TYPES )... )
          )
```

graph_pattern

```
MATCH path_pattern_list [ graph_pattern_where_clause ]
```

graph_pattern_quantifier

```
{ fixed_quantifier | general_quantifier }
```

graph_pattern_variable

```
identifier
```

graph_pattern_variable_declaration

```
graph_pattern_variable
```

graph_pattern_where_clause

```
WHERE search_condition
```

graphql_query_for_DV

```
root_query_field
```

graph_reference

```
graph_name [ graph_ref_as_of_clause ]
```

graph_ref_as_of_clause

```
( AS OF (
      ( ( SCN | TIMESTAMP ) expr )
      |
      ( PERIOD FOR valid_time_column expr ) )
  )
```

graph_table_column_definition

```
value_expression [ AS column_name ]
```

graph_table_columns_clause

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

graph_table

```
GRAPH_TABLE ( graph_reference graph_pattern graph_table_shape )
```

graph_table_label_and_properties

```
[ graph_table_label_properties_clause ]
 [  ( graph_table_label_clause ) ... ]
```

graph_table_label_clause

```
(( ( LABEL label_identifier ) | (DEFAULT LABEL) )) [ graph_table_label_properties_clause ]
```

graph_table_label_properties_clause

```
( NO PROPERTIE) | ( PROPERTIES graph_table_properties_alternatives ) )
```

graph_table_properties_alternatives

```
[ ARE ] ALL COLUMNS [ EXCEPT  ( column_name_list ) ]
  | ( column_or_expression ... )
```

graph_table_rows_clause

```
{ ONE ROW PER MATCH | one_row_per_iteration }
```

graph_table_shape

```
[ graph_table_rows_clause ] graph_table_columns_clause
```

group_by_clause

```
[ GROUP BY
   { expr
   | c_alias
   | position
   | rollup_cube_clause
   | grouping_sets_clause
   }
     [, { expr
        | c_alias
        | position
        | 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 ]
```

hier_ancestor_expression

```
HIER_ANCESTOR ( member_expression AT
                 { LEVEL level_ref
                  | DEPTH depth_expression
                 }
               )
```

hier_ancestors

```
HIER_ANCESTORS ( member_expr [ self_clause ] )
```

hier_descendants

```
HIER_DESCENDANTS ( member_expr [ self_clause ]
 [ AT { LEVEL level_name | DEPTH level_depth | LEAF } ])
```

hier_attr_clause

```
hier_attr_name [ classification_clause ]...
```

hier_attr_name

```
{   MEMBER_NAME
  | MEMBER_UNIQUE_NAME
  | MEMBER_CAPTION
  | MEMBER_DESCRIPTION
  | LEVEL_NAME
  | HIER_ORDER
  | DEPTH
  | IS_LEAF
  | PARENT_LEVEL_NAME
  | PARENT_UNIQUE_NAME
}
```

hier_attrs_clause

```
HIERARCHICAL ATTRIBUTES ( hier_attr_clause [, hier_attr_clause ]... )
```

hier_children

```
HIER_CHILDREN ( member_expr  )
```

hier_condition

```
HIER_CONDITION ( member_expr IS [ NOT ]
                 [ IN member_set |
                 { PARENT | CHILD | ROOT ANCESTOR | [LEAF] DESCENDANT
                  | SIBLING | ANCESTOR AT LEVEL level }
                  [ OR SELF] OF ] WITHIN HIERARCHY hierarchy_ref
                  [ { INCLUDE | SKIP } WHEN NULL]
               )
```

hier_count

```
HIER_COUNT ( [DISTINCT] member_set
    WITHIN HIERARCHY hierarchy_ref
    [ { INCLUDE | SKIP } WHEN NULL ] )
```

hier_distinct

```
HIER_DISTINCT ( member_set )
```

hier_expand

```
HIER_EXPAND ( member_set BY member_to_set_func )
```

hier_first_expression

```
 HIER_FIRST ( member_set )
```

hier_id

```
 MEASURES | ( ( dim_alias.) hier_alias )
```

hier_ids

```
 hier_id [ hier_id ]...
```

hier_last_expression

```
 HIER_LAST ( member_set )
```

hier_intersect

```
HIER_INTERSECT ( member_set , member_set )
```

hier_lead_lag_clause

```
member_expression  OFFSET offset_expr
  [ WITHIN
    {  { LEVEL | PARENT }
    | ACROSS ANCESTOR AT LEVEL level_ref [ POSITION FROM { BEGINNING | END } ]
    }
  ]
```

hier_lead_lag_expression

```
{ HIER_LEAD | HIER_LAG } ( hier_lead_lag_clause )
```

hier_level_members

```
HIER_LEVEL_MEMBERS
  ( member_expr [ WITHIN { LEVEL | PARENT | ANCESTOR AT
                           { LEVEL level_name | DEPTH level_depth }
                         }
                ]
  )
```

hier_member_at_expression

```
 HIER_MEMBER_AT ( member_set , number )
```

hier_member_set

```
HIER_MEMBER_SET ( member_expr [ , member_expr ...] )
```

hier_minus

```
HIER_UNION ( member_set , member_set )
```

hier_navigation_expression

```
{
  hier_ancestor_expression
  | hier_parent_expression
  | hier_lead_lag_expression
  | hier_first_expression
  | hier_last_expression
  | hier_member_at_expression
}
```

hier_parent_expression

```
HIER_PARENT ( member_expression )
```

hier_siblings

```
HIER_SIBLINGSS ( member_expr [ self_clause ] )
```

hier_position

```
HIER_POSITION ( member_expr IN member_set
  WITHIN HIERARCHY hierarchy_ref
  [ { INCLUDE | SKIP } WHEN NULL ] )
```

hier_range

```
HIER_RANGE ( member_set
             { FIRST number | LAST number | BETWEEN number AND number }
             [ PERCENT ]
           )
```

hier_ref

```
[ schema. ] hier_name [ [ AS ] hier_alias ] [ DEFAULT ]
```

hier_union

```
HIER_UNION ( member_set , member_set )
```

hier_union_all

```
HIER_UNION_ALL ( member_set , member_set )
```

hier_using_clause

```
USING [ schema. ] attribute_dimension level_hier_clause
```

hier_window

```
HIER_WINDOW ( member_set RELATIVE TO member_expr
  BETWEEN { preceding_boundary | following_boundary } )
```

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 ]
)
```

hierarchy_ref

```
[ attr_dim_alias. ] hier_alias
```

identity_clause

```
 GENERATED [ ALWAYS | BY DEFAULT [ ON NULL [ FOR ( INSERT { ONLY | AND UPDATE })]]]
 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_compression_policy

```
{ table_compression { SEGMENT | GROUP }
  { { AFTER ilm_time_period OF { { NO ACCESS } | { NO MODIFICATION } | CREATION } }
  | { ON function_name } }
```

```
}
|
{ { ROW STORE COMPRESS ADVANCED
  | COLUMN STORE COMPRESS FOR QUERY
  }
  ROW AFTER ilm_time_period OF NO MODIFICATION
}
```

ilm_inmemory_policy

```
{ SET INMEMORY [ inmemory_attributes ]
| MODIFY INMEMORY inmemory_memcompress
| NO INMEMORY
}
[ SEGMENT ]
{ AFTER ilm_time_period OF { NO ACCESS | NO MODIFICATION | CREATION }
        | ON function_name
        }
```

ilm_policy_clause

```
{ ilm_compression_policy | ilm_tiering_policy | ilm_inmemory_policy }
```

ilm_tiering_policy

```
{ TIER TO tablespace [ SEGMENT | GROUP ] [ ON function_name ] }
|
{ TIER TO tablespace READ ONLY [ SEGMENT | GROUP ]
  { { AFTER ilm_time_period OF { { NO ACCESS } | { NO MODIFICATION } | CREATION } }
  | { ON function_name } } }
```

ilm_time_period

```
integer { { DAY | DAYS } | { MONTH | MONTHS } | { YEAR | YEARS } }
```

implementation_clause

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

immutable_table_clauses

```
immutable_table_no_drop_clause immutable_table_no_delete_clause
```

immutable_table_no_delete_clause

```
NO DELETE ( [ LOCKED ] | ( UNTIL integer DAYS AFTER INSERT [LOCKED] ) )
```

immutable_table_no_drop_clause

```
NO DROP ( [ LOCKED ] | ( UNTIL integer DAYS AFTER INSERT [LOCKED] ) )
```

import_keys

```
IMPORT [ ENCRYPTION ] KEYS WITH SECRET secret
  FROM 'filename'
  [ FORCE KEYSTORE ]
  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
  | annotations_clause
  }...
]
```

index_compression

```
{ prefix_compression
| advanced_index_compression
}
```

index_expr

```
{ column | column_expression }
```

index_ilm_clause

```
ILM
    (
      [  ADD  POLICY | ( DELETE POLICY  policy_name ) ]
         policy_clause
    )
```

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
                | 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] [read_only_clause] [indexing_clause] [partitioning_storage_clause]
  [, PARTITION [partition] [read_only_clause] [indexing_clause] [partitioning_storage_clause]]... )
```

individual_hash_subparts

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

inline_constraint

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

inline_external_table

```
 EXTERNAL '(' '(' column_definition ',' ')' inline_external_table_properties ')'
```

inline_external_table_properties

```
 TYPE [ access_driver_type ] external_table_data_props
  [ REJECT LIMIT { integer | UNLIMITED }
```

inline_ref_constraint

```
{ SCOPE  IS [ schema. ] scope_table
| WITH ROWID
```

```
| [ CONSTRAINT constraint_name ]
  references_clause
  [ constraint_state ]
}
```

inmemory_attributes

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

inmemory_clause

```
( INMEMORY [ inmemory_attributes ] [TEXT ( ( "column_name")/","
           | ("column_name" USING "policy_name")/"," ) ] )
| NO INMEMORY
```

inmemory_column_clause

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

inmemory_distribute

```
DISTRIBUTE [ AUTO | BY { ROWID RANGE | PARTITION | SUBPARTITION } ]
           [ FOR SERVICE { DEFAULT | ALL | service_name | NONE } ]
```

inmemory_duplicate

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

inmemory_memcompress

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

inmemory_priority

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

inmemory_table_clause

```
[ { INMEMORY [ inmemory_attributes ] } | { NO INMEMORY } ]
[ inmemory_column_clause ]
```

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 ]...) ]
```

insert_op

```
INSERT pathExpr = rhsExpr
     [ { IGNORE | ERROR | REPLACE } ON EXISTING ]
```

```
    [ { IGNORE | ERROR | REMOVE | NULL } ON NULL ]
    [ { IGNORE | ERROR | NULL } ON EMPTY ]
    [ { IGNORE | ERROR } ON ERROR ]
```

insert_values_clause

```
VALUES ( expr | DEFAULT [, expr | DEFAULT ]...)
            [,( expr | DEFAULT [, expr | DEFAULT ]... )]+
```

instance_clauses

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

instances_clause

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

integer

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

intersect_op

```
INTERSECT pathExpr = rhsExpr
    [ { IGNORE | ERROR | CREATE | NULL } ON MISSING ]
    [ { IGNORE | ERROR | NULL } ON NULL ]
```

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 }
```

is_json_args

```
 { ([ FORMAT JSON ][ STRICT | LAX ] [ { ALLOW  | DISALLOW } SCALARS ]
    [ { WITH | WITHOUT } UNIQUE KEYS ] )
  |
  ( VALIDATE [CAST] [USING] schema )
}
```

is_label_declaration

```
IS label_expression
```

isolate_keystore

```
ISOLATE KEYSTORE INDENTIFIED BY isolated_keystore_password
FROM ROOT KEYSTORE [ FORCE KEYSTORE ]
IDENTIFIED BY { EXTERNAL STORE | united_keystore_password }
[ WITH BACKUP [ USING 'backup_identifier' ] ]
```

join_clause

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

join_path_clause

```
JOIN PATH join_path_name ON join_condition
```

JSON_array_content

```
    { JSON_ARRAY_enumeration_content | JSON_ARRAY_query_content }
```

JSON_array_element

```
expr [ format_clause ]
```

JSON_array_enumeration_content

```
    ( , [ JSON_ARRAY_element ] ... )
    [ JSON_on_null_clause ] [ JSON_returning_clause ]
    [ STRICT ]
```

JSON_Collection_table

```
[ WITH ETAG ]
[ ( expression_column [ , expression_column ...]
     , constraint [ , constraint ...] ) ]
[ physical_properties ] [ table_properties ]
```

JSON_column_modifier

```
VALUE
  | ARRAY
  | OBJECT
  | SCALAR [ JSON_scalar_modifier ]
```

JSON_column_definition

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

JSON_columns_clause

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

JSON_exists_column

```
column_name [ JSON_value_return_type ]
EXISTS [ PATH ] [ JSON_path ] [ JSON_exists_on_error_clause ]
[ JSON_exists_on_empty_clause ]
```

JSON_exists_on_empty_clause

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

JSON_exists_on_error_clause

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

JSON_modifier_list

```
  ( JSON_column_modifier [, JSON_column_modifier ]...)
   | "JSON_column_modifier"
```

JSON_nested_path

```
NESTED [ PATH ] JSON_path  JSON_columns_clause
```

JSON_object_content

```
( "*" | [ entry ] ... )
    [ JSON_on_null_clause ] [ JSON_returning_clause ]
    [ STRICT ]
    [ WITH UNIQUE KEYS ]
```

JSON_on_null_clause

```
{ NULL | ABSENT } ON NULL
```

JSON_parameters

```
   ( TABLESPACE tablespace
   | storage_clause
   | ( (CHUNK | PCTVERSION | FREEPOOLS) integer )
   | RETENTION
   ) ...
```

JSON_passing_clause

```
PASSING expr AS identifier [, expr AS identifier ]...
```

JSON_path

```
JSON_basic_path_expression | JSON_relative_object_access
```

json_property_graph_object_access_expr

```
element_variable.property_name . (json_prop_graph_obj_access_step)
. [ json_item_method ]
```

json_prop_graph_obj_access_step

```
{ json_object_key | array_step }
```

JSON_query_column

```
column_name JSON_query_return_type FORMAT JSON
  [ (ALLOW | DISALLOW) SCALARS ] [ JSON_query_wrapper_clause ]
  PATH JSON_path [ JSON_query_on_error_clause ]
```

JSON_query_on_empty_clause

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

JSON_query_on_error_clause

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

JSON_query_on_mismatch_clause

```
 ( ERROR | NULL ) ON MISMATCH
```

JSON_query_quotes_clause

```
 { KEEP | OMIT } QUOTES [ ON SCALAR STRING ]
```

JSON_query_return_type

```
 { VARCHAR2 [ ( size [BYTE | CHAR] ) ]
| CLOB [ reference | value ]
| BLOB [ reference | value ]
| JSON
| VECTOR }
```

JSON_query_returning_clause

```
[ RETURNING JSON_query_return_type ][ { ALLOW | DISALLOW } SCALARS ]
[ PRETTY ] [ ASCII ]
```

JSON_query_wrapper_clause

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

JSON_relative_object_access

```
JSON_object_key [ array_step ]
 ( "." JSON_object_key [ array_step ] )...
```

JSON_returning_clause

```
RETURNING VARCHAR2 { [ ( size [BYTE | CHAR] ) ]
   [ WITH TYPENAME ]
| CLOB [ reference | value ]
| BLOB [ reference | value ]
| JSON
}
```

JSON_scalar_modifier

```
NUMBER
  | STRING
  | BINARY_DOUBLE
  | BINARY_FLOAT
  | DATE
  | TIMESTAMP [ WITH TIME ZONE ]
  | NULL
  | BOOLEAN
  | BINARY
  | INTERVAL { YEAR TO MONTH | DAY TO SECOND }
```

JSON_storage_clause

```
  JSON ( json_column … ) STORE AS
  ( ( ( json_parameters )
     | [ LOB_segname ] [ ( json_parameters )]
    ) )
```

JSON_table_on_empty_clause

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

JSON_table_on_error_clause

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

JSON_transform_returning_clause

```
RETURNING {
     VARCHAR2 [ ( size [BYTE | CHAR] ) ]
     | CLOB [ reference | value ]
     | BLOB [ reference | value ]
     | JSON
     | BOOLEAN
     | VECTOR
    }
```

JSON_type_column

```
JSON ( JSON_column_modifier [ , JSON_column_modifier ]...)
```

JSON_value_column

```
column_name [ JSON_value_return_type ] [ TRUNCATE ]
  [ PATH ] [ JSON_path ] [ JSON_value_on_error_clause ]
  [ JSON_value_on_empty_clause ]
  [ JSON_value_on_mismatch_clause ]
```

JSON_value_mapper_clause

```
USING CASE-SENSITIVE MAPPING
```

JSON_value_on_empty_clause

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

JSON_value_on_error_clause

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

JSON_value_on_mismatch_clause

```
JSON_value_on_mismatch (
   ( IGNORE | ERROR | NULL )
    ON MISMATCH
   [ (  (MISSING DATA) | (EXTRA DATA) | (TYPE ERROR)  ) ]
  ) ...
```

JSON_value_return_object_instance

```
object_type_name [ JSON_value_mapper_clause ]
```

JSON_value_return_type

```
{ VARCHAR2 [ ( size [BYTE | CHAR] ) TRUNCATE ]
| CLOB
| ( NUMBER [ ( precision [, scale] ) ]
| { ALLOW | DISALLOW } [ BOOLEAN ] TO NUMBER [CONVERSION] )
| DATE [ { TRUNCATE | PRESERVE } TIME ]
| TIMESTAMP [ TIMESTAMP WITH TIME ZONE ]
| BOOLEAN
| SDO_GEOMETRY
| JSON_value_return_object_instance
| VECTOR
 }
```

JSON_value_returning_clause

```
 RETURNING JSON_value_return_type  [ ASCII ]
```

keep_op

```
KEEP ( pathExpr ) [ , pathExpr ]...
  [ { IGNORE | ERROR } ON MISSING ]
```

key_clause

```
KEY { [(] attribute [)] | ( attribute [, attribute]... ) }
```

key_value_clause

```
[ KEY ] string ( : | IS )
  { ( column_name ) [ column_tags_clause ]
     | ( duality_view_subquery ) | [ duality_view_subquery ]
  }
  | UNNEST ( duality_view_subquery }
```

key_management_clauses

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

keystore_clause

```
KEYSTORE IDENTIFIED BY [( EXTERNAL STORE ) | keystore_password ]
            [ NO REKEY ]
```

keystore_management_clauses

```
{ create_keystore
| open_keystore
| close_keystore
| backup_keystore
| alter_keystore_password
| merge_into_new_keystore
| merge_into_existing_keystore
| isolate_keystore
| unite_keystore
}
```

label

```
identifier
```

label_disjunction

```
{ label_expression | label }
```

label_expression

```
{ label | label_disjunction }
```

lead_lag_clause

```
HIERARCHY hierarchy_ref OFFSET offset_expr
  [ {
      WITHIN { LEVEL | PARENT }
    | ACROSS ANCESTOR AT LEVEL level_ref [ POSITION FROM { BEGINNING | END }
    }
  ]
```

lead_lag_expression

```
lead_lag_function_name ( calc_meas_expression ) OVER ( lead_lag_clause )
```

lead_lag_function_name

```
{ LAG | LAG_DIFF | LAG_DIFF_PERCENT | LEAD | LEAD_DIFF | LEAD_DIFF_PERCENT }
```

letter

```
{ upper_letter | lower_letter }
```

level_clause

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

level_group_type

```
DYNAMIC | MATERIALIZED [ USING [ schema.] table ]
```

level_hier_clause

```
( level [ CHILD  OF level ]... )
```

level_member_literal

```
level_ref { pos_member_keys | named_member_keys }
```

level_specification

```
( [ [ dim_name. ] hier_name. ] level_name )
```

levels_clause

```
 LEVELS ([ level_specification ]...) level_group_type
```

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 [, column]... )
[ AUTOMATIC [ STORE IN ( tablespace [, tablespace ]... ) ] ]
(PARTITION [ partition ]
    list_values_clause table_partition_description
  [, PARTITION [ partition ]
        list_values_clause table_partition_description
        [ external_part_subpart_data_props ]
  ]...
)
```

list_partitionset_clause

```
PARTITIONSET BY LIST (column)
  PARTITION BY CONSISTENT HASH (column [, column]...)
  [ SUBPARTITION BY { { RANGE | HASH } (column [, column]...)
                    | LIST (column)
                    }
  [ subpartition_template ]
  ]
  PARTITIONS AUTO ( list_partitionset_desc [, list_partitionset_desc]... )
```

list_partitionset_desc

```
PARTITIONSET partition_set list_values_clause
  [ TABLESPACE SET tablespace_set ]
  [ LOB_storage_clause ]
  [ SUBPARTITIONS STORE IN ( tablespace_set … ) ]
```

list_subpartition_desc

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

list_values

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

list_values_clause

```
VALUES ( list_values | DEFAULT )
```

listagg_overflow_clause

```
{ ON OVERFLOW ERROR }
|
{ ON OVERFLOW TRUNCATE 'truncation-indicator' [ { WITH | WITHOUT } COUNT ] }
```

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 SET tablespace_set ) ]
  | ( TABLESPACE tablespace | TABLESPACE SET tablespace_set )
  ]
```

lob_rename_parameters

```
[ PARTITION | SUBPARTITION ] old_segment_name TO new_segment_name
```

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 | TABLESPACE SET tablespace_set }
  | LOB_parameters [storage_clause]
  }...
| storage_clause
```

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 ] ]...
    )
  ]
```

lockdown_features

```
{ DISABLE | ENABLE } FEATURE
{ { = ( 'feature' [, 'feature' ]... ) }
| { ALL [ EXCEPT = ( 'feature' [, 'feature' ]... ) ] }
}
```

lockdown_options

```
{ DISABLE | ENABLE } OPTION
{ { = ( 'option' [, 'option' ]... ) }
| { ALL [ EXCEPT = ( 'option' [, 'option' ]... ) ] }
}
```

lockdown_statements

```
{ DISABLE | ENABLE } STATEMENT
{ { = ( 'SQL_statement' [, 'SQL_statement' ]... ) }
| { = ( 'SQL_statement' ) statement_clauses }
| { ALL [ EXCEPT = ( 'SQL_statement' [, 'SQL_statement' ]... ) ] }
}
```

logfile_clause

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

logfile_clauses

```
{ { ARCHIVELOG [ MANUAL ]
  | NOARCHIVELOG
```

```
  }
| [ NO ] FORCE LOGGING
| SET STANDBY NOLOGGING FOR {DATA AVAILABILITY | LOAD PERFORMANCE}
| 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'
}
```

logical_replication_clause

```
{
             DISABLE LOGICAL REPLICATION
             | ENABLE LOGICAL REPLICATION [ ( { ALL KEYS | ALLOW NOVALIDATE KEYS }
             | ([ NO ] PARTIAL JSON ) )... ]
            }
```

logging_clause

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

lower_bound

```
( unsigned_integer )
```

lower_case_name

```
 { lower_letter | _ } [name_continue_lower]...
```

lower_letter

```
 { a | ... | z }
```

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
      | USING INSTANCES { ALL | integer }
      | 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 }
```

meas_aggregate_clause

```
AGGREGATE BY aggr_function
```

measure_ref

```
[ MEASURES. ] meas_name
```

measures_clause

```
MEASURES  ( av_measure [, av_measure]... )
```

member_expression

```
{ level_member_literal
  | hier_navigation_expression
  | CURRENT MEMBER
  | NULL
  | ALL
}
```

member_set

```
 { member_to_set_func
              | set_to_set_func
              | hier_member_set }
```

member_to_set_func

```
hier_ancestors | hier_descendants | hier_siblings
 | hier_children  | hier_level_members
```

memoptimize_read_clause

```
[ { (MEMOPTIMIZE FOR READ) | (NO MEMOPTIMIZE FOR READ) } ]
```

memoptimize_write_clause

```
[ { (MEMOPTIMIZE FOR WRITE) | (NO MEMOPTIMIZE FOR WRITE) } ]
```

merge_insert_clause

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

merge_into_existing_keystore

```
MERGE KEYSTORE 'keystore1_location' [ IDENTIFIED BY keystore1_password ]
  INTO EXISTING KEYSTORE 'keystore2_location'
  IDENTIFIED BY { EXTERNAL STORE | 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
    { EXTERNAL STORE | keystore3_password }
```

merge_op

```
MERGE pathExpr = rhsExpr
    [ { IGNORE | ERROR | CREATE | NULL } ON MISSING ]
    [ { IGNORE | ERROR } ON MISMATCH ]
    [ { IGNORE | ERROR | NULL } ON NULL ]
    [ { IGNORE | ERROR } ON EMPTY ]
```

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 ]
   [ filter_condition ]
   [ dependent_tables_clause ]
   [ update_index_clauses ]
   [ parallel_clause ]
   [ ONLINE ]
   [ 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
          }
   ]
   [ filter_condition ]
   [ dependent_tables_clause ]
   [ update_index_clauses ]
   [ parallel_clause ]
   [ ONLINE ]
   [ allow_disallow_clustering ]
```

merge_update_clause

```
WHEN MATCHED THEN
UPDATE SET column = { expr | DEFAULT }
```

```
          [, column = { expr | DEFAULT } ]...
[ where_clause ]
[ DELETE where_clause ]
```

migrate_key

```
{ USE | SET } [ ENCRYPTION ] KEY 'key_id'
  IDENTIFIED BY HSM_auth_string
  [ FORCE KEYSTORE ]
  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 ]
       }
    ]...
}
```

minus_op

```
MINUS pathExpr = rhsExpr
    [ { IGNORE | ERROR | CREATE | NULL } ON MISSING ]
    [ { IGNORE | ERROR | NULL } ON NULL ]
```

model_clause

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

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
  ]...
)
```

modified_external_table

```
 EXTERNAL MODIFY modify_external_table_properties
```

modify_col_properties

```
column [ datatype ] [ RESERVABLE | NOT RESERVABLE ]
       [ DOMAIN [ domain_owner ] . domain_name ]
       [ COLLATE column_collation_name ]
       [ default_clause | identity_clause | DROP IDENTITY ]
       [ { ENCRYPT encryption_spec } | DECRYPT ]
       [ inline_constraint ... ]
       [ LOB_storage_clause ]
       [ alter_XMLSchema_clause ]
       [ annotations_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_domain
}
```

modify_external_table_properties

```
DEFAULT DIRECTORY directory
 [ LOCATION '(' directory ':' ''' location_specifier ''' ')' ]
 [ ACCESS PARAMETERS
   [ BADFILE filename ]
   [ LOGFILE filename ]
   [ DISCARDFILE filename ] ]
 [ REJECT LIMIT { integer | UNLIMITED ]
```

modify_filegroup_clause

```
MODIFY FILEGROUP filegroup_name
  SET '[ file_type. ] property_name' = 'property_value'
```

modify_hash_partition

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

```
  | read_only_clause
  | 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 ] [ parallel_clause ]
| 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 ( list_values )
  | { add_range_subpartition
    | add_list_subpartition
    | add_hash_subpartition
    }
  | coalesce_table_subpartition
  | [ REBUILD ] UNUSABLE LOCAL INDEXES
  | read_only_clause
  | 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_partitionset

```
MODIFY PARTITIONSET partition_set ADD VALUES ( ( value )[, value]...)
```

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
   | read_only_clause
   | indexing_clause
   }
```

modify_table_default_attrs

```
MODIFY DEFAULT ATTRIBUTES
   [ FOR partition_extended_name ]
   [ deferred_segment_creation ]
   [ read_only_clause ]
   [ 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 ( list_values )
| read_only_clause
```

```
| indexing_clause
}
```

modify_to_partitioned

```
MODIFY [ table_partitioning_clauses | NONPARTITIONED ]
  [ filter_condition ]
  [ ONLINE ]
  [ UPDATE INDEXES
    [ ( index { local_partitioned_index | global_partitioned_index | GLOBAL }
        [, index { local_partitioned_index | global_partitioned_index | GLOBAL } ]... )
    ]
  ]
```

modify_virtcol_properties

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

modify_volume_clause

```
MODIFY VOLUME asm_volume
   [ MOUNTPATH 'mountpath_name' ]
  [ USAGE 'usage_name' ]
```

modify_table_default_attrs

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

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_partitionset

```
MOVE PARTITIONSET partition_set TABLESPACE SET tablespace_set
   [ SUBPARTITIONS STORE IN ( (tablespace_set)[ ,"tablespace_set ]... )  ]
   [ LOB ( (lob_column_name)[ , lob_column_name ]...)  STORE AS ( TABLESPACE SET
   ( (tablespace_set)[ , tablespace_set]... ) ]
```

move_table_clause

```
MOVE
   [ filter_condition ]
   [ ONLINE ]
   [ segment_attributes_clause ]
```

```
   [ table_compression ]
   [ index_org_table_clause ]
   [ { lob_storage_clause | json_storage_clause | varray_col_properties }... ]
   [ parallel_clause ]
   [ allow_disallow_clustering ]
   [ UPDATE INDEXES
     [ ( index { segment_attributes_clause
               | update_index_partition }
         [, index { segment_attributes_clause
                  | update_index_partition } ]...
       )
     ]
   ]
```

move_table_partition

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

move_table_subpartition

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

move_to_filegroup_clause

```
MOVE FILE 'ASM_filename' TO FILEGROUP filegroup_name
```

move_keys

```
MOVE [ENCRYPTION] KEYS
    TO NEW KEYSTORE keystore_location1
    IDENTIFIED BY keystore1_password
    FROM [FORCE] KEYSTORE
    IDENTIFIED BY keystore_password
    [WITH IDENTIFIER IN
      { 'key_identifier' [, 'key_identifier']... | ( subquery ) } ]
    [WITH BACKUP [USING 'backup_identifier'] ];
```

multi_column_for_loop

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

multi_table_insert

```
{ ALL
  { insert_into_clause [ insert_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
                                     }
      }
```

name

```
name_start [name_continue]...
```

name_continue

```
{ letter | digit | _ }
```

name_continue_lower

```
 { lower_letter | digit | _ }
```

name_start

```
{ letter | _ }
```

named_member_keys

```
'['  attr_name = [, attr_name = member_key_expr ]... ']'
```

nested_clause

```
table_reference (NESTED [PATH]) identifier
[
("." [ JSON_object_key array_step ] ) |
("," JSON_basic_path_expression )
]
[ JSON_table_on_error_clause ]
[ JSON_table_on_empty_clause ]
 JSON_columns_clause
```

nested_path_op

```
NESTED PATH path_expr ( [ [ operation ] [ , operation ]...] )
```

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_field_name

```
 object_type_name_lower
```

object_field_name_lower

```
 [schema_name_lower .] simple_object_type_name_lower
```

object_gen_clause

```
JSON { (key_value_clause) [, (key_value_clause)...]
      [ , flex_clause ]  }
```

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 ]
       ]...
)
```

one_row_per_iteration

```
{ ONE ROW PER VERTEX ( vertex_variable )
  | ONE ROW PER STEP ( vertex_variable, edge_variable, vertex_variable)
}
```

on_error_clause

```
 ( ERROR | NULL ) ON ERROR
```

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
  [ FORCE KEYSTORE ]
  IDENTIFIED BY { EXTERNAL STORE | keystore_password }
  [ CONTAINER = { ALL | CURRENT } ]
```

operation

```
  add_set_op
  | append_op
  | case_op
  | copy_op
  | insert_op
  | intersect_op
  | keep_op
  | merge_op
  | minus_op
  | nested_path_op
  | prepend_op
  | remove_op
  | remove_set_op
  | rename_op
  | replace_op
  | set_op
  | sort_op
  | union_op
```

option_values

```
{ { VALUE = ( 'option_value' [, 'option_value' ]... ) }
  |
  { MINVALUE = 'option_value' }
  |
  { MAXVALUE = 'option_value' }
}...
```

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 }
   [ constraint_state ]
| CHECK (condition) [ constraint_state ] [ precheck_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
     ]...
    [, 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 ] }
```

parallel_pdb_creation_clause

```
PARALLEL [ integer ]
```

parenthesized_path_pattern_expression

```
( path_pattern_expression [ parenthesized_path_pattern_where_clause ]   )
```

parenthesized_path_pattern_where_clause

```
WHERE search_condition
```

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

partitionset_clauses

```
{ range_partitionset_clause | list_partitionset_clause }
```

password_parameters

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

patch_common

```
target_expr [ json_query_returning_clause ] [ pretty ]
 [ ASCII ] [ TRUNCATE ] [ json_query_on_error_clause ]
```

path_concatenation

```
 path_term path_factor
```

path_factor

```
{ path_primary | quantifier_path_primary }
```

path_pattern

```
path_pattern_expression
```

path_pattern_expression

```
path_term
```

path_pattern_list

```
path_pattern [ , path_pattern ]...
```

path_prefix_clause

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

path_primary

```
{ element_pattern | parenthesized_path_pattern_expression }
```

path_term

```
{ path_factor | path_concatenation }
```

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 }
| SET STANDBY NOLOGGING FOR {DATA AVAILABILITY | LOAD PERFORMANCE}
```

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_managed_recovery

```
RECOVER MANAGED STANDBY DATABASE [ CANCEL ]
```

pdb_open

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

pdb_recovery_clauses

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

pdb_refresh_mode_clause

```
REFRESH MODE { MANUAL | EVERY refresh_interval { MINUTES | HOURS} | NONE }
```

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_refresh_mode_clause
  | REFRESH [ pdb_refresh_switchover_clause ]
  | PRIORITY value
  | SET CONTAINER_MAP = 'map_object'
  }
}
| CONTAINERS { DEFAULT TARGET = { (container_name) | NONE
              |  HOST "=" "'" "hostname" "'"
        |  PORT "=" "number" }
              }
```

pdb_storage_clause

```
STORAGE
  { ( { MAXSIZE { UNLIMITED | size_clause }
        |
        MAX_AUDIT_SIZE { UNLIMITED | size_clause }
        |
        MAX_DIAG_SIZE { UNLIMITED | size_clause }
      }...
    )
  |
  UNLIMITED
  }
```

pdb_snapshot_clause

```
ENABLE SNAPSHOT { MANUAL | EVERY snapshot_interval { HOURS | MINUTES } | NONE}
```

pdb_unplug_clause

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

period_definition

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

permanent_tablespace_attrs

```
{ MINIMUM EXTENT size_clause
| BLOCKSIZE integer [ K ]
| logging_clause
| FORCE LOGGING
| tablespace_encryption_clause
| default_tablespace_params
| { ONLINE | OFFLINE }
| extent_management_clause
| segment_management_clause
| flashback_mode_clause
| lost_write_protection
}...
```

permanent_tablespace_clause

```
TABLESPACE [ IF NOT EXISTS ] tablespace
  [ DATAFILE file_specification [, file_specification ]... ]
  [ permanent_tablespace_attrs ]
```

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 PARTITION ATTRIBUTES external_table_clause [ REJECT LIMIT ]
  }
| 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 }...
```

policy_clause

```
 ( [ OPTIMIZE condition_clause ]  | tiering_clause [ PLSQL_function_name ] )
```

pos_member_keys

```
'['  member_key_expr [, member_key_expr]...']'
```

preceding_boundary

```
{ UNBOUNDED PRECEDING | offset_expr PRECEDING }
AND
{ CURRENT MEMBER
  | offset_expr  { PRECEDING | FOLLOWING }
  | UNBOUNDED FOLLOWING
}
```

prefix_compression

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

prepare_clause

```
   PREPARE MIRROR COPY copy_name
   [ WITH { EXTERNAL | NORMAL | HIGH } REDUNDANCY ]
   [ FOR DATABASE target_cdb_name ]
```

prepend_op

```
PREPEND pathExpr = rhsExpr
    [ { IGNORE | ERROR | CREATE | NULL } ON MISSING ]
    [ { IGNORE | ERROR | REPLACE | CREATE } ON MISMATCH ]
    [ { IGNORE | ERROR | NULL } ON NULL ]
    [ { IGNORE | ERROR } ON EMPTY ]
```

privilege_audit_clause

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

program_unit

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

property_clause

```
PROPERTY { SET | REMOVE } DEFAULT_CREDENTIAL = SYSTEM.OPCTEST
```

proxy_clause

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

qdr_expression

```
QUALIFY ( calc_meas_expression, qualifier )
```

qualified_disk_clause

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

qualified_template_clause

```
ATTRIBUTE
( redundancy_clause
  striping_clause
)
```

qualifier

```
hierarchy_ref = member_expression
```

quantifier_path_primary

```
{ path_primary | graph_pattern_quantifier }
```

query_block

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

query_object_field_name

```
 object_field_name
```

query_partition_clause

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

query_rewrite_clause

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

query_scalar_field_name

```
 { name | quoted_name }
```

query_table_expression

```
{ query_name
| [ schema. ]
  { table [ modified_external_table
          | partition_extension_clause
          | @ dblink
          ]
  | { view | materialized view } [ @ dblink ]
  | hierarchy
  | analytic_view [ HIERARCHIES
    ( [ [ attr_dim. ] hierarchy [, [ attr_dim. ] hierarchy ]... ] ) ]
  | inline_external_table
  } [sample_clause]
```

```
| [ LATERAL ] (subquery [ subquery_restriction_clause ])
| table_collection_expression
}
```

quiesce_clauses

```
QUIESCE RESTRICTED | UNQUIESCE
```

quotagroup_clauses

```
{ ADD QUOTAGROUP quotagroup_name [ SET property_name = property_value ]
| MODIFY QUOTAGROUP quotagroup_name SET property_name = property_value
| MOVE FILEGROUP filegroup_name TO quotagroup_name
| DROP QUOTAGROUP quotagroup_name
}
```

quoted_name

```
 " any_char "
```

qry_transform_clause

```
ENABLE QUERY TRANSFORM [ RELY | NORELY ]
```

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
        [ external_part_subpart_data_props ]
      ]...
)
```

range_partitionset_clause

```
PARTITIONSET BY RANGE (column [, column]...)
  PARTITION BY CONSISTENT HASH (column [, column]...)
  [ SUBPARTITION BY { { RANGE | HASH } (column [, column]...)
                    | LIST (column)
                    }
  [ subpartition_template ]
  ]
  PARTITIONS AUTO ( range_partitionset_desc [, range_partitionset_desc]... )
```

range_partitionset_desc

```
PARTITIONSET partition_set range_values_clause
  [ TABLESPACE SET tablespace_set ]
  [ LOB_storage_clause ]
  [ SUBPARTITIONS STORE IN ( tablespace_set … ) ]
```

range_subpartition_desc

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

range_values_clause

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

read_only_clause

```
{ READ ONLY } | { READ WRITE }
```

rebalance_diskgroup_clause

```
REBALANCE
  [ { [ { WITH | WITHOUT } phase [, phase]... ] [ POWER integer ] [ WAIT | NOWAIT ] }
    |
    { MODIFY POWER [ integer ] }
  ]
```

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 | PARITY | DOUBLE]
```

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 ]
```

regular_entry

```
[ KEY ] expr VALUE expr
                       | expr [ ":" expr ]
```

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) ]
[ immutable_table_clauses ]
[ blockchain_table_clauses ]
[ DEFAULT COLLATION collation_name ]
[ ON COMMIT { DROP | PRESERVE } DEFINITION ]
[ ON COMMIT { DELETE | PRESERVE } ROWS ]
[ physical_properties ]
[ table_properties ]
```

relocate_clause

```
RELOCATE [ TO 'instance_name' ]
| NORELOCATE
```

remove_op

```
REMOVE pathExpr [ { IGNORE | ERROR } ON MISSING ]
```

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_lob_storage_clause

```
RENAME LOB lob_item lob_rename_parameters
```

rename_op

```
RENAME pathExpr WITH stringLiteral [ { IGNORE | ERROR } ) ON MISSING ]
```

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 ]
```

replace_op

```
REPLACE pathExpr = rhsExpr
    [ { CREATE | IGNORE | ERROR} ON MISSING ]
    [ {  IGNORE | ERROR | REMOVE | NULL } ON NULL ]
    [ { IGNORE | ERROR | NULL } ON EMPTY ]
    [ { IGNORE | ERROR  } ON ERROR ]
```

resize_disk_clause

```
RESIZE ALL [ 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 }
}
```

result_cache_clause

```
RESULT_CACHE ( (( [ MODE {DEFAULT | FORCE} ] [ , STANDBY {ENABLE | DISABLE} ] )
                 | ( [ STANDBY {ENABLE | DISABLE} ] [ , MODE {DEFAULT | FORCE} ] ) ) )
```

return_rows_clause

```
RETURN { UPDATED | ALL } ROWS
```

returning_clause

```
{ RETURN | RETURNING } ( [ OLD | NEW ] expr [ , [ OLD | NEW ] expr ]... )
INTO  data_item [ , data_item ]...
```

reverse_migrate_key

```
SET [ ENCRYPTION ] KEY
  IDENTIFIED BY software_keystore_password
  [ FORCE KEYSTORE ]
  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 } ]...
```

rhs_expr

```
{ (sql_expr [ FORMAT JSON ] ) | ( PATH path_expr ) }
```

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)
```

root_query_field

```
root_query_field_name [ directives ] selection_set
```

root_query_field_name

```
object_type_name_lower
```

routine_clause

```
[ schema. ] [ type. | package. ]
{ function | procedure | method }
[ @dblink_name ]
( [ argument [, argument ]... ] )
```

row_limiting_clause

```
[ OFFSET offset { ROW | ROWS } ]
[ fetch_clause
  row_limiting_partition_clause
  row_specification
  accuracy_clause
]
```

row_limiting_partition_clause

```
[( partition_count [ { PARTITION | PARTITIONS } BY ] partition_by_expr , )... ]
```

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
```

row_specification

```
[ rowcount | percent PERCENT  ] { ROW | ROWS } { ONLY | WITH TIES }
```

sample_clause

```
SAMPLE [ BLOCK ]
       (sample_percent)
       [ SEED (seed_value) ]
```

scalar_field_name

```
{ lower_case_name | quoted_name }
```

schema_name_lower

```
  { lower_case_name | quoted_name }
```

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 ]
  [ STOP ]
```

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
| add_update_secret_seps
| delete_secret_seps
}
```

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 | TABLESPACE SET tablespace_set }
| logging_clause
}...
```

segment_management_clause

```
SEGMENT SPACE MANAGEMENT { AUTO | MANUAL }
```

self_clause

```
{ INCLUDE | EXCLUDE } SELF
```

select_list

```
{ *
| { query_name.*
  | [ schema. ] { table | view | materialized view } .*
  | t_alias.*
  | expr [ [ AS ] c_alias ]
  }
    [, { query_name.*
       | [ schema. ] { table | view | materialized view } .*
       | t_alias.*
       | expr [ [ AS ] c_alias ]
       }
    ]...
}
```

selection

```
{ field | fragment_spread }
```

selection_list

```
selection ...
```

selection_set

```
( [ {  selection_list } ] ) | ( {  selection_list } )
```

service_name_convert

```
SERVICE_NAME_CONVERT =
  { ( 'service_name', 'replacement_service_name'
      [, 'service_name', 'replacement_service_name' ]... )
    |
    NONE
  }
```

services_clause

```
SERVICES = { ( 'service_name' [, 'service_name' ]... )
             | ALL [ EXCEPT ( 'service_name' [, 'service_name' ]... ) ] }
```

set_encryption_key

```
{ SET ENCRYPTION KEY
  IDENTIFIED BY
    {
        "wallet_password"
         {  WITH BACKUP USING backup_identifier
              |  REVERSE MIGRATE USING  "OKV_password" WITH BACKUP
                 [ USING backup_identifier ]
            }
     |  "OKV_password"  MIGRATE USING  "wallet_password"
    }
}
```

set_key

```
SET [ ENCRYPTION ] KEY { mkid:mk | mk }
  [ USING TAG 'tag' ]
```

```
  [ USING ALGORITHM 'encrypt_algorithm' ]
  [ FORCE KEYSTORE ]
  IDENTIFIED BY { EXTERNAL STORE | keystore_password }
  [ WITH BACKUP [ USING 'backup_identifier' ] ]
  [ CONTAINER = { ALL | CURRENT } ]
```

set_key_tag

```
SET TAG 'tag' FOR 'key_id'
  [ FORCE KEYSTORE ]
  IDENTIFIED BY { EXTERNAL STORE | keystore_password }
  [ WITH BACKUP [ USING 'backup_identifier' ] ]
```

set_op

```
SET pathExpr = rhsExpr
    [ {IGNORE | ERROR | REPLACE} ON EXISTING ]
    [ { IGNORE | ERROR | CREATE } ON MISSING ]
    [ { IGNORE | ERROR | REMOVE | NULL } ON NULL ]
    [ {  IGNORE | ERROR | NULL } ON EMPTY ]
    [ { IGNORE | ERROR} ON ERROR ]
```

set_parameter_clause

```
parameter_name =
   parameter_value [, parameter_value ]...
   [ COMMENT = string ]
   [ DEFERRED ]
   [ CONTAINER = { CURRENT | ALL } ]
   [ { SCOPE = { MEMORY | SPFILE | BOTH }
     | SID = { 'sid' | '*' }
     }...
   ]
```

set_to_set_function

```
hier_union | hier_union_all | hier_intersect |
  hier_minus | hier_distinct | hier_range
  | hier_window | hier_expand
```

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 }'
```

shards_clause

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

share_clause

```
HIERARCHY hierarchy_ref
  { PARENT
  | LEVEL level_ref
```

```
  | MEMBER member_expression
  }
```

share_of_expression

```
SHARE_OF ( calc_meas_expression  share_clause )
```

sharing_clause

```
SHARING = { METADATA | DATA | NONE }
```

shrink_clause

```
SHRINK SPACE [ COMPACT ] [ CASCADE ]
```

shutdown_dispatcher_clause

```
SHUTDOWN [ IMMEDIATE ] dispatcher_name
```

simple_case_expression

```
expr
  { WHEN comparison_expr THEN return_expr }...
```

simple_expression

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

simple_object_type_name_lower

```
  { lower_case_name | quoted_name }
```

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
{ insert_values_clause [ returning_clause ]
| subquery
} [ error_logging_clause ]
```

size_clause

```
integer [ K | M | G | T | P | E ]
```

sort_op

```
SORT path_expr
   ( REVERSE
     | ( [ REMOVE NULLS ] ORDER BY ( path_expr [ ASC | DESC ] )
            [ , ( path_expr [ ASC | DESC ] ) ]...
     | [ ASC | DESC ] [ UNIQUE ] [ REMOVE NULLS ]
       )
    )
  [ {  IGNORE | ERROR | NULL } ON MISSING ]
  [ {  IGNORE | ERROR | NULL } ON MISMATCH ]
  [ {  IGNORE | ERROR } ON EMPTY ]
  [ {  IGNORE | ERROR } ON ERROR ]
```

source_clause

```
[ schema . ] fact_table_or_view [ REMOTE ] ( [ [ AS ] alias ] )
```

source_file_directory

```
SOURCE_FILE_DIRECTORY = { 'directory_path_name' | NONE }
```

source_file_name_convert

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

source_predicate

```
vertex_reference IS [NOT] SOURCE OF edge_reference
```

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
  ( nested_table_partition_spec, nested_table_partition_spec
    [split_nested_table_part]
  ) [split_nested_table_part]
```

split_partitionset

```
SPLIT PARTITIONSET "partition_set"
   INTO
     { range_partitionset_clause | list_partitionset_clause } ,
        PARTITIONSET partition_set
         [ TABLESPACE SET tablespace_set"]
       [ LOB ((lob_column_name)[, lob_column_name]...) STORE AS
       ( TABLESPACE SET ( (tablespace_set)[ , tablespace_set]... ) ]
          [ SUBPARTITIONS STORE IN ( (tablespace_set)[ , tablespace_set]... )  ]
```

split_table_partition

```
SPLIT partition_extended_name
  { AT (literal [, literal]... )
    [ INTO ( range_partition_desc, range_partition_desc ) ]
  | VALUES ( list_values )
    [ 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 ]
    [ filter_condition ]
    [ dependent_tables_clause ]
    [ update_index_clauses ]
    [ parallel_clause ]
    [ allow_disallow_clustering ]
    [ ONLINE ]
```

split_table_subpartition

```
SPLIT subpartition_extended_name
  { AT ( literal [, literal]... )
    [ INTO ( range_subpartition_desc, range_subpartition_desc ) ]
  | VALUES ( list_values )
    [ INTO ( list_subpartition_desc, list_subpartition_desc ) ]
  | INTO ( { range_subpartition_desc [, range_subpartition_desc ]...
           | list_subpartition_desc [, list_subpartition_desc ]... }
         , subpartition_spec )
  } [ filter_condition ]
    [ dependent_tables_clause ]
    [ update_index_clauses ]
    [ parallel_clause ]
    [ allow_disallow_clustering ]
    [ ONLINE ]
```

sql_format

```
[+ | -] days hours : minutes : seconds [. frac_secs ]
```

standard_actions

```
ACTIONS
  { { object_action | ALL } [ ( column [ , column )... ]
    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 = { ( 'cdb_name' [, 'cdb_name' ]... )
           | { ALL [ EXCEPT ( 'cdb_name' [, 'cdb_name' ]... ) ] }
           | NONE
           }
```

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
  }
}
```

statement_clauses

```
CLAUSE
{ { = ( 'clause' [, 'clause' ]... ) }
| { = ( 'clause' ) clause_options }
| { ALL [ EXCEPT = ( 'clause' [, 'clause' ]... ) ] }
}
```

static_base_profile

```
FROM base_profile
```

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 ]
```

sub_av_clause

```
USING [ schema . ] base_av_name [ hierarchies_clause ]
  [ filter_clauses] [ add_meas_clause ]
```

subpartition_by_hash

```
SUBPARTITION BY HASH (column [, column ]...)
   [ SUBPARTITIONS integer
        [ STORE IN (tablespace [, tablespace ]...) ]
   | subpartition_template
   ]
```

subpartition_by_list

```
SUBPARTITION BY LIST ( column [, 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 } [ ALL ] subquery
    [ { UNION [ALL] | INTERSECT | MINUS } [ ALL ] 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_subset_replication_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_subset_replication_clause

```
 DATA SUBSET DATABASE 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

```
 [ { INTERNAL | EXTERNAL } ]
[ deferred_segment_creation ]
[ read_only_clause ]
[ indexing_clause ]
[ segment_attributes_clause ]
[ table_compression | prefix_compression ]
[ inmemory_clause ]
[ ilm_clause ]
```

```
[ OVERFLOW [ segment_attributes_clause ] ]
[ { json_storage_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
| consistent_hash_partitions
| directory_based_partitions
| composite_directory_based_partitions
| consistent_hash_with_subpartitions
| partitionset_clauses
}
```

table_properties

```
[ column_properties ]
[ read_only_clause ]
[ indexing_clause ]
[ table_partitioning_clauses ]
[ attribute_clustering_clause ]
[ CACHE | NOCACHE ]
[ result_cache_clause ]
[ parallel_clause ]
[ ROWDEPENDENCIES | NOROWDEPENDENCIES ]
[ enable_disable_clause ]...
[ row_movement_clause ]
[ logical_replication_clause ]
[ flashback_archive_clause ]
[ ROW ARCHIVAL ]
[ { AS subquery } | { FOR EXCHANGE WITH TABLE [ schema .] table } ]
[ [ FOR STAGING ]
```

table_reference

```
{ { { ONLY (query_table_expression) | query_table_expression }
  [ flashback_query_clause ]
  [ pivot_clause | unpivot_clause | row_pattern_clause ] }
| containers_clause
| shards_clause
}
[ t_alias ]
| values_clause
```

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_clause

```
ENCRYPTION [ { [ tablespace_encryption_spec ] ENCRYPT } | DECRYPT ]
```

tablespace_encryption_spec

```
USING 'encrypt_algorithm' MODE 'cipher_mode'
```

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 }
}
```

table_tags_clause

```
WITH
  {
  { CHECK | NOCHECK } [ ETAG ]
  | INSERT
  | NOINSERT
  | UPDATE
  | NOUPDATE
  | DELETE
  | NODELETE
  }...
```

tempfile_reuse_clause

```
TEMPFILE REUSE
```

temporary_tablespace_clause

```
{ { TEMPORARY TABLESPACE [ IF NOT EXISTS ] }
| { LOCAL TEMPORARY TABLESPACE FOR { ALL | LEAF } }
} tablespace
[ TEMPFILE file_specification [, file_specification ]... ]
[ tablespace_group_clause ]
[ extent_management_clause ]
[ tablespace_encryption_clause ]
```

tiering_clause

```
 SEGMENT TIER TO LOW_COST_TBS
```

timebucket_optional_clause

```
  ON OVERFLOW ROUND
```

```
    | ON OVERFLOW ERROR
    | LAST DAY OF MONTH
```

timeout_clause

```
DROP AFTER integer { M | H }
```

trace_file_clause

```
TRACE
  [ AS 'filename' [ REUSE ] ]
  [ RESETLOGS | NORESETLOGS ]
```

tracking_statistics_clause

```
 AFTER time_interval
   ( DAYS
   | MONTHS
   | YEARS )
    OF [ NO ] ( ACCESS | MODIFICATION | CREATION )
```

truncate_partition_subpart

```
TRUNCATE { partition_extended_names | subpartition_extended_names }
   [ { DROP [ ALL ] | REUSE } STORAGE ]
   [ update_index_clauses [ parallel_clause ] ] [ CASCADE ]
```

ts_file_name_convert

```
FILE_NAME_CONVERT =
  ( 'filename_pattern', 'replacement_filename_pattern'
      [, 'filename_pattern', 'replacement_filename_pattern' ]... )
  [ KEEP ]
```

undo_mode_clause

```
LOCAL UNDO { ON | OFF }
```

undo_tablespace

```
  [ BIGFILE | SMALLFILE ]
UNDO TABLESPACE tablespace
  [ DATAFILE file_specification [, file_specification ]...]
```

undo_tablespace_clause

```
UNDO TABLESPACE [ IF NOT EXISTS ] tablespace
  [ DATAFILE file_specification [, file_specification ]... ]
  [ extent_management_clause ]
  [ tablespace_retention_clause ]
  [ tablespace_encryption_clause ]
```

undrop_disk_clause

```
UNDROP DISKS
```

union_op

```
UNION pathExpr = rhsExpr
    [ { IGNORE | ERROR | CREATE | NULL } ON MISSING ]
    [ { IGNORE | ERROR | NULL } ON NULL ]
```

unite_keystore

```
UNITE KEYSTORE INDENTIFIED BY isolated_keystore_password
WITH ROOT KEYSTORE [ FORCE KEYSTORE ]
```

```
IDENTIFIED BY { EXTERNAL STORE | united_keystore_password }
[ WITH BACKUP [ USING 'backup_identifier' ] ]
```

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 ]
```

upper_bound

```
( unsigned_integer )
```

upper_letter

```
{ A | ... | Z }
```

use_key

```
USE [ ENCRYPTION ] KEY 'key_id'
  [ USING TAG 'tag' ]
  [ FORCE KEYSTORE ]
  IDENTIFIED BY { EXTERNAL STORE | 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' ]... )
  | ALL [ EXCEPT ( 'tablespace' [, 'tablespace' ]... ) ]
  | NONE
  }
  [ SNAPSHOT COPY | NO DATA | COPY | MOVE | NOCOPY ]
```

usergroup_clauses

```
{ ADD USERGROUP 'usergroup' WITH MEMBER 'user' [, 'user']...
| MODIFY USERGROUP 'usergroup' { ADD | DROP } MEMBER 'user' [, 'user']...
| DROP USERGROUP 'usergroup'
}
```

using_clause

```
USING [ schema. ] fact_table_or_view  [ [ AS ] alias ]
```

using_function_clause

```
USING [ schema. ] [ package. | type. ] function_name
```

using_index_clause

```
USING INDEX
  { [ schema. ] index
  | (create_index_statement)
  | index_properties
  }
```

using_snapshot_clause

```
USING SNAPSHOT { snapshot_name | AT SCN snapshot_SCN | AT snapshot_timestamp }
```

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 [ , expr ]... ) [ , ( expr [ , expr]...) ] )
  [ [AS] t_alias ( c_alias [, c_alias ]... ) ]
       )
```

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
}
```

vector_index_organization_clause

```
ORGANIZATION { INMEMORY [ NEIGHBOR ] GRAPH |  [ NEIGHBOR ] PARTITIONS  }
   [ WITH ] [ DISTANCE metric name ]
```

vector_index_parameters_clause

```
[ PARAMETERS (  { vector_index_parameters_hnsw_clause
            | vector_index_parameters_ivf_clause
           }
        )
]
```

vector_index_parameters_hnsw_clause

```
TYPE HNSW , { [ NEIGHBORS ] | M } max_closest_vectors_connected
  , [ EFCONSTRUCTION ] max_candidates_to_consider
```

vector_index_parameters_ivf_clause

```
TYPE IVF , {  NEIGHBOR PARTITIONS number_of_partitions
            | [ SAMPLES_PER_PARTITION number_of_samples ]
```

```
            | [ MIN_VECTORS_PER_PARTITION min_number_of_vectors_per_partition]
           }
```

vertex_pattern

```
 ( optional_element_pattern_filter )
```

vertex_tables_clause

```
VERTEX TABLES ( ( vertex_table_definition )... )
```

vertex_tables_definition

```
graph_element_name_and_key [ graph_table_label_and_properties ]
```

vertex_table_reference

```
 graph_element_name |
    graph_element_key  REFERENCES graph_element_name ( column_name_list )
```

virtual_column_definition

```
column [ datatype [ COLLATE column_collation_name ] ]
  [ VISIBLE | INVISIBLE ]
  [ GENERATED ALWAYS ] AS (column_expression) [ VIRTUAL | MATERIALIZED ]
  [ evaluation_edition_clause ] [ unusable_editions_clause ]
  [ inline_constraint [ inline_constraint ]... ]
```

where_clause

```
WHERE condition
```

wildcard

```
[ id "." ] id "." "*"
```

window_clause

```
WINDOW  window_name AS ( window_specification )
              [ , window_name AS ( window_specification )] ...
```

window_expression

```
aggregate_function OVER ( window_clause )
```

windowing_clause

```
{ ROWS | RANGE | GROUPS}
{ 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
  }
}
[ EXCLUDE CURRENT ROW
 | EXCLUDE GROUPS
```

```
 | EXCLUDE TIES
 | EXCLUDE NO OTHERS ]
```

window_specification

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

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 } ] ]...
  )
```

XMLnamespaces_clause

```
XMLNAMESPACES
  ( { string AS identifier } | { DEFAULT string }
     [, { 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 | [ [ NOT ] TRANSPORTABLE ] 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 ] ]
```

zero_downtime_software_patching_clauses

```
SWITCHOVER LIBRARY path FOR ALL CONTAINERS
```

zonemap_attributes

```
{ TABLESPACE tablespace
| SCALE 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
- Oracle Built-In Data Types
- Oracle-Supplied Data Types
- Converting to Oracle Data Types

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 **Built-In Data Type Summary** table lists the built-in data types available. Oracle Database uses a code to identify the data type internally. This is the number in the **Code** column of the **Built-In Data Type Summary** table. You can verify the codes in the table using the `DUMP` function.

In addition to the built-in data types listed in the **Built-In Data Type Summary** table, Oracle Database uses many data types internally that are visible via 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.
119	JSON	Maximum size is 32 megabytes.
252	BOOLEAN	The BOOLEAN data type comprises the distinct truth values *True* and *False*. Unless prohibited by a NOT NULL constraint, the boolean data type also supports the truth value *UNKOWN* as the null value.
127	VECTOR	The VECTOR data type represents a vector as a series of numbers stored in one of the following formats: • INT8 (8-bit integers) • FLOAT32 (32-bit floating-point numbers) • FLOAT64 (64-bit floating-point numbers) FLOAT32 and FLOAT64 are IEEE standards. Oracle Database automatically casts the values as needed.

> **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 }
```

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(38)`

`FLOAT` (**Note 2**)
`DOUBLE PRECISION` (**Note 3**)
`REAL` (**Note 4**)
	`FLOAT(126)`
	`FLOAT(126)`
	`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.

Table 7-1 (Cont.) Number Format Elements

Element	Example	Description
. (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.
S	`S9999` `9999S`	Returns negative value with a leading minus sign (-). Returns positive value with a leading plus sign (+). 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.

Table 7-1 (Cont.) Number Format Elements

Element	Example	Description
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`	No	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.

Table 7-2 (Cont.) Datetime Format Elements

Element	TO_* datetime functions?	Description
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).
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	No	Week of year (1-52 or 1-53) based on the ISO standard.
IYY IY I	No	Last 3, 2, or 1 digit(s) of ISO year.
IYYY	No	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).

Table 7-2 (Cont.) Datetime Format Elements

Element	TO_* datetime functions?	Description
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	No	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).
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	No	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.

Table 7-2 (Cont.) Datetime Format Elements

Element	TO_* datetime functions?	Description
W	No	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	No	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 Basic SQL*Plus Commands are SQL*Plus commands available in the command-line interface. Not all commands or command parameters are shown.

> **See Also:**
>
> *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*]

Table A-1 (Cont.) *Basic SQL*Plus Commands*

Database Operation	SQL*Plus Command
Show SQL*Plus system variables or environment settings	`SHOW { ALL | ERRORS | USER | system_variable [, system_variable] ...}`
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]`

Table A-1 (Cont.) *Basic SQL*Plus Commands*

Database Operation	SQL*Plus Command
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*]
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

Table A-1 (Cont.) *Basic SQL*Plus Commands*

Database Operation	SQL*Plus Command
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

Symbols

@ (at sign) SQL*Plus command, *A-3*
/ (slash) SQL*Plus command, *A-4*

A

ABS function, *2-1*
ACOS function, *2-1*
action_audit_clause, *5-1*
activate_standby_db_clause, *5-1*
add_binding_clause, *5-1*
add_column_clause, *5-1*
add_disk_clause, *5-1*
add_filegroup_clause, *5-1*
add_hash_index_partition, *5-1*
add_hash_partition_clause, *5-1*
add_hash_subpartition, *5-1*
add_list_partition_clause, *5-1*
add_list_subpartition, *5-1*
add_logfile_clauses, *5-1*
ADD_MONTHS function, *2-1*
add_mv_log_column_clause, *5-1*
add_overflow_clause, *5-1*
add_period_clause, *5-1*
add_range_partition_clause, *5-1*
add_range_subpartition, *5-1*
add_system_partition_clause, *5-1*
add_table_partition, *5-1*
add_update_secret, *5-1*
add_volume_clause, *5-1*
ADMINISTER KEY MANAGEMENT statement, *1-1*
advanced_index_compression, *5-1*
aggregate functions, *2-1*
alias_file_name, *5-1*
all_clause, *5-1*
allocate_extent_clause, *5-1*
allow_disallow_clustering, *5-1*
ALTER ANALYTIC VIEW statement, *1-1*
ALTER ATTRIBUTE DIMENSION statement, *1-1*
ALTER AUDIT POLICY statement, *1-1*
ALTER CLUSTER statement, *1-1*
ALTER DATABASE LINK statement, *1-1*
ALTER DATABASE statement, *1-1*
ALTER DIMENSION statement, *1-1*
ALTER DISKGROUP statement, *1-1*
ALTER FLASHBACK ARCHIVE statement, *1-1*
ALTER FUNCTION statement, *1-1*
ALTER HIERARCHY statement, *1-1*
ALTER INDEX statement, *1-1*
ALTER INDEXTYPE statement, *1-1*
ALTER INMEMORY JOIN GROUP statement, *1-1*
ALTER JAVA statement, *1-1*
ALTER LIBRARY statement, *1-1*
ALTER LOCKDOWN PROFILE statement, *1-1*
ALTER MATERIALIZED VIEW LOG statement, *1-1*
ALTER MATERIALIZED VIEW statement, *1-1*
ALTER MATERIALIZED ZONEMAP statement, *1-1*
ALTER OPERATOR statement, *1-1*
ALTER OUTLINE statement, *1-1*
ALTER PACKAGE statement, *1-1*
ALTER PLUGGABLE DATABASE statement, *1-1*
ALTER PROCEDURE statement, *1-1*
ALTER PROFILE statement, *1-1*
ALTER PROPERTY GRAPH statement, *1-1*
ALTER RESOURCE COST statement, *1-1*
ALTER ROLE statement, *1-1*
ALTER ROLLBACK SEGMENT statement, *1-1*
ALTER SEQUENCE statement, *1-1*
ALTER SESSION statement, *1-1*
ALTER SYNONYM statement, *1-1*
ALTER SYSTEM statement, *1-1*
ALTER TABLE statement, *1-1*
ALTER TABLESPACE SET statement, *1-1*
ALTER TABLESPACE statement, *1-1*
ALTER TRIGGER statement, *1-1*
ALTER TYPE statement, *1-1*
ALTER USER statement, *1-1*
ALTER VIEW statement, *1-1*
alter_automatic_partitioning, *5-1*
alter_datafile_clause, *5-1*
alter_external_table, *5-1*
alter_index_partitioning, *5-1*
alter_interval_partitioning, *5-1*
alter_iot_clauses, *5-1*
alter_keystore_password, *5-1*
alter_mapping_table_clauses, *5-1*
alter_mv_refresh, *5-1*
alter_overflow_clause, *5-1*

alter_query_rewrite_clause, *5-1*
alter_session_set_clause, *5-1*
alter_system_reset_clause, *5-1*
alter_system_set_clause, *5-1*
alter_table_partitioning, *5-1*
alter_table_properties, *5-1*
alter_tablespace_attrs, *5-1*
alter_tablespace_encryption, *5-1*
alter_tempfile_clause, *5-1*
alter_varray_col_properties, *5-1*
alter_XMLSchema_clause, *5-1*
alter_zonemap_attributes, *5-1*
alternate_key_clause, *5-1*
American National Standards Institute (ANSI)
converting to Oracle data types, *6-6*
analytic functions, *2-1*
analytic_clause, *5-1*
ANALYZE statement, *1-1*
ANSI-supported data types, *6-1*
any_types, *6-5*
APPEND SQL*Plus command, *A-3*
APPENDCHILDXML function, *2-1*
application_clauses, *5-1*
APPROX_COUNT_DISTINCT function, *2-1*
APPROX_COUNT_DISTINCT_AGG function, *2-1*
APPROX_COUNT_DISTINCT_DETAIL function, *2-1*
APPROX_MEDIAN function, *2-1*
APPROX_PERCENTILE function, *2-1*
APPROX_PERCENTILE_AGG function, *2-1*
APPROX_PERCENTILE_DETAIL function, *2-1*
archive_log_clause, *5-1*
array_DML_clause, *5-1*
array_step, *5-1*
ASCII function, *2-1*
ASCIISTR function, *2-1*
ASIN function, *2-1*
ASM_filename, *5-1*
ASSOCIATE STATISTICS statement, *1-1*
ATAN function, *2-1*
ATAN2 function, *2-1*
attr_dim_attributes_clause, *5-1*
attr_dim_level_clause, *5-1*
attr_dim_using_clause, *5-1*
attribute_clause, *5-1*
attribute_clustering_clause, *5-1*
attributes_clause, *5-1*
AUDIT (Unified Auditing) statement, *1-1*
audit_operation_clause, *5-1*
audit_schema_object_clause, *5-1*
auditing_by_clause, *5-1*
auditing_on_clause, *5-1*
autoextend_clause, *5-1*
av_meas_expression, *5-1*
av_measure, *5-1*
av_simple_expression, *5-1*
AVG function, *2-1*

B

backup_keystore, *5-1*
base_measure_clause, *5-1*
BETWEEN condition, *4-1*
BFILENAME function, *2-1*
BIN_TO_NUM function, *2-1*
binding_clause, *5-1*
BITAND function, *2-1*
bitmap_join_index_clause, *5-1*
build_clause, *5-1*
built-in data types, *6-1*, *6-2*
by_users_with_roles, *5-1*

C

cache_clause, *5-1*
cache_specification, *5-1*
calc_meas_order_by_clause, *5-1*
calc_measure_clause, *5-1*
calculated measure expressions, *3-1*
CALL statement, *1-1*
CARDINALITY function, *2-1*
CASE expressions, *3-1*
CAST function, *2-1*
CEIL function, *2-1*
cell_assignment, *5-1*
cell_reference_options, *5-1*
CHANGE SQL*Plus command, *A-3*
character_datatypes, *6-2*
character_set_clause, *5-1*
CHARTOROWID function, *2-1*
check_datafiles_clause, *5-1*
check_diskgroup_clause, *5-1*
checkpoint_clause, *5-1*
CHR function, *2-1*
classification_clause, *5-1*
clause_options, *5-1*
clear_free_space_clause, *5-1*
close_keystore, *5-1*
cluster_clause, *5-1*
CLUSTER_DETAILS (analytic) function, *2-1*
CLUSTER_DETAILS function, *2-1*
CLUSTER_DISTANCE (analytic) function, *2-1*
CLUSTER_DISTANCE function, *2-1*
CLUSTER_ID (analytic) function, *2-1*
CLUSTER_ID function, *2-1*
cluster_index_clause, *5-1*
CLUSTER_PROBABILITY (analytic) function, *2-1*
CLUSTER_PROBABILITY function, *2-1*
cluster_range_partitions, *5-1*
CLUSTER_SET (analytic) function, *2-1*
CLUSTER_SET function, *2-1*
clustering_column_group, *5-1*

clustering_columns, *5-1*
clustering_join, *5-1*
clustering_when, *5-1*
COALESCE function, *2-1*
coalesce_index_partition, *5-1*
coalesce_table_partition, *5-1*
coalesce_table_subpartition, *5-1*
COLLATION function, *2-1*
COLLECT function, *2-1*
column expressions, *3-1*
column_association, *5-1*
column_clauses, *5-1*
column_definition, *5-1*
column_properties, *5-1*
COMMENT statement, *1-1*
COMMIT statement, *1-1*
commit_switchover_clause, *5-1*
component_actions, *5-1*
COMPOSE function, *2-1*
composite_hash_partitions, *5-1*
composite_list_partitions, *5-1*
composite_range_partitions, *5-1*
compound conditions, *4-1*
compound expressions, *3-1*
CON_DBID_TO_ID function, *2-1*
CON_GUID_TO_ID function, *2-1*
CON_NAME_TO_ID function, *2-1*
CON_UID_TO_ID function, *2-1*
CONCAT function, *2-1*
conditional_insert_clause, *5-1*
conditions, *4-1*
    see also SQL conditions, *4-1*
CONNECT SQL*Plus command, *A-3*
consistent_hash_partitions, *5-1*
consistent_hash_with_subpartitions, *5-1*
constraint, *5-1*
constraint_clauses, *5-1*
constraint_state, *5-1*
container_data_clause, *5-1*
containers_clause, *5-1*
context_clause, *5-1*
controlfile_clauses, *5-1*
CONVERT function, *2-1*
convert_database_clause, *5-1*
convert_redundancy_clause, *5-1*
converting to Oracle data types, *6-6*
CORR function, *2-1*
CORR_K function, *2-1*
CORR_S function, *2-1*
COS function, *2-1*
COSH function, *2-1*
cost_matrix_clause, *5-1*
COUNT function, *2-1*
COVAR_POP function, *2-1*
COVAR_SAMP function, *2-1*
CREATE ANALYTIC VIEW statement, *1-1*
CREATE ATTRIBUTE DIMENSION statement, *1-1*
CREATE AUDIT POLICY statement, *1-1*
CREATE CLUSTER statement, *1-1*
CREATE CONTEXT statement, *1-1*
CREATE CONTROLFILE statement, *1-1*
CREATE DATABASE LINK statement, *1-1*
CREATE DATABASE statement, *1-1*
CREATE DIMENSION statement, *1-1*
CREATE DIRECTORY statement, *1-1*
CREATE DISKGROUP statement, *1-1*
CREATE EDITION statement, *1-1*
CREATE FLASHBACK ARCHIVE statement, *1-1*
CREATE FUNCTION statement, *1-1*
CREATE HIERARCHY statement, *1-1*
CREATE INDEX statement, *1-1*
CREATE INDEXTYPE statement, *1-1*
CREATE INMEMORY JOIN GROUP statement, *1-1*
CREATE JAVA statement, *1-1*
CREATE LIBRARY statement, *1-1*
CREATE LOCKDOWN PROFILE statement, *1-1*
CREATE LOGICAL PARTITION TRACKING statement, *1-1*
CREATE MATERIALIZED VIEW LOG statement, *1-1*
CREATE MATERIALIZED VIEW statement, *1-1*
CREATE MATERIALIZED ZONEMAP statement, *1-1*
CREATE OPERATOR statement, *1-1*
CREATE OUTLINE statement, *1-1*
CREATE PACKAGE BODY statement, *1-1*
CREATE PACKAGE statement, *1-1*
CREATE PFILE statement, *1-1*
CREATE PLUGGABLE DATABASE statement, *1-1*
CREATE PROCEDURE statement, *1-1*
CREATE PROFILE statement, *1-1*
CREATE PROPERTY GRAPH statement, *1-1*
CREATE RESTORE POINT statement, *1-1*
CREATE ROLE statement, *1-1*
CREATE ROLLBACK SEGMENT statement, *1-1*
CREATE SCHEMA statement, *1-1*
CREATE SEQUENCE statement, *1-1*
CREATE SPFILE statement, *1-1*
CREATE SYNONYM statement, *1-1*
CREATE TABLE statement, *1-1*
CREATE TABLESPACE SET statement, *1-1*
CREATE TABLESPACE statement, *1-1*
CREATE TRIGGER statement, *1-1*
CREATE TYPE BODY statement, *1-1*
CREATE TYPE statement, *1-1*
CREATE USER statement, *1-1*
CREATE VECTOR INDEX statement, *1-1*
CREATE VIEW statement, *1-1*
create_datafile_clause, *5-1*

create_file_dest_clause, *5-1*
create_key, *5-1*
create_keystore, *5-1*
create_mv_refresh, *5-1*
create_pdb_clone, *5-1*
create_pdb_from_seed, *5-1*
create_pdb_from_xml, *5-1*
create_zonemap_as_subquery, *5-1*
create_zonemap_on_table, *5-1*
cross_outer_apply_clause, *5-1*
CUBE_TABLE function, *2-1*
CUME_DIST (aggregate) function, *2-1*
CUME_DIST (analytic) function, *2-1*
currency
    group separators, *7-2*
currency symbol
    ISO, *7-2*
    local, *7-2*
    union, *7-3*
CURRENT_DATE function, *2-1*
CURRENT_TIMESTAMP function, *2-1*
CURSOR expressions, *3-1*
CV function, *2-1*
cycle_clause, *5-1*

D

data types
    ANSI-supported, *6-1*
    converting to Oracle, *6-6*
    Oracle built-in, *6-1*, *6-2*
    Oracle-supplied, *6-1*, *6-5*
    overview, *6-1*
    user-defined, *6-1*
database_file_clauses, *5-1*
database_logging_clauses, *5-1*
datafile_tempfile_clauses, *5-1*
datafile_tempfile_spec, *5-1*
DATAOBJ_TO_MAT_PARTITION function, *2-1*
DATAOBJ_TO_PARTITION function, *2-1*
date format models, *7-3*, *7-4*
    long, *7-4*
    short, *7-5*
datetime expressions, *3-1*
datetime_datatypes, *6-2*
db_user_proxy_clauses, *5-1*
DB2 data types
    restrictions on, *6-6*
dblink, *5-1*
dblink_authentication, *5-1*
DBTIMEZONE function, *2-1*
deallocate_unused_clause, *5-1*
decimal characters
    specifying, *7-2*
DECODE function, *2-1*
DECOMPOSE function, *2-1*
default_aggregate_clause, *5-1*
default_cost_clause, *5-1*
default_index_compression, *5-1*
default_measure_clause, *5-1*
default_selectivity_clause, *5-1*
default_settings_clauses, *5-1*
default_table_compression, *5-1*
default_tablespace, *5-1*
default_tablespace_params, *5-1*
default_temp_tablespace, *5-1*
deferred_segment_creation, *5-1*
DEL SQL*Plus command, *A-3*
DELETE statement, *1-1*
delete_secret, *5-1*
DENSE_RANK (aggregate) function, *2-1*
DENSE_RANK (analytic) function, *2-1*
dependent_tables_clause, *5-1*
DEPTH function, *2-1*
DEREF function, *2-1*
DESCRIBE SQL*Plus command, *A-3*
dim_by_clause, *5-1*
dim_key, *5-1*
dim_order_clause, *5-1*
dim_ref, *5-1*
dimension_join_clause, *5-1*
DISASSOCIATE STATISTICS statement, *1-1*
DISCONNECT SQL*Plus command, *A-4*
disk_offline_clause, *5-1*
disk_online_clause, *5-1*
diskgroup_alias_clauses, *5-1*
diskgroup_attributes, *5-1*
diskgroup_availability, *5-1*
diskgroup_directory_clauses, *5-1*
diskgroup_template_clauses, *5-1*
diskgroup_volume_clauses, *5-1*
distributed_recov_clauses, *5-1*
dml_table_expression_clause, *5-1*
domain_index_clause, *5-1*
DROP ANALYTIC VIEW statement, *1-1*
DROP ATTRIBUTE DIMENSION statement, *1-1*
DROP AUDIT POLICY statement, *1-1*
DROP CLUSTER statement, *1-1*
DROP CONTEXT statement, *1-1*
DROP DATABASE LINK statement, *1-1*
DROP DATABASE statement, *1-1*
DROP DIMENSION statement, *1-1*
DROP DIRECTORY statement, *1-1*
DROP DISKGROUP statement, *1-1*
DROP EDITION statement, *1-1*
DROP FLASHBACK ARCHIVE statement, *1-1*
DROP FUNCTION statement, *1-1*
DROP HIERARCHY statement, *1-1*
DROP INDEX statement, *1-1*
DROP INDEXTYPE statement, *1-1*
DROP INMEMORY JOIN GROUP statement, *1-1*
DROP JAVA statement, *1-1*

DROP LIBRARY statement, *1-1*
DROP LOCKDOWN PROFILE statement, *1-1*
DROP MATERIALIZED VIEW LOG statement, *1-1*
DROP MATERIALIZED VIEW statement, *1-1*
DROP MATERIALIZED ZONEMAP statement, *1-1*
DROP OPERATOR statement, *1-1*
DROP OUTLINE statement, *1-1*
DROP PACKAGE statement, *1-1*
DROP PLUGGABLE DATABASE statement, *1-1*
DROP PROCEDURE statement, *1-1*
DROP PROFILE statement, *1-1*
DROP PROPERTY GRAPH statement, *1-1*
DROP RESTORE POINT statement, *1-1*
DROP ROLE statement, *1-1*
DROP ROLLBACK SEGMENT statement, *1-1*
DROP SEQUENCE statement, *1-1*
DROP SYNONYM statement, *1-1*
DROP TABLE statement, *1-1*
DROP TABLESPACE SET statement, *1-1*
DROP TABLESPACE statement, *1-1*
DROP TRIGGER statement, *1-1*
DROP TYPE BODY statement, *1-1*
DROP TYPE statement, *1-1*
DROP USER statement, *1-1*
DROP VIEW statement, *1-1*
drop_binding_clause, *5-1*
drop_column_clause, *5-1*
drop_constraint_clause, *5-1*
drop_disk_clause, *5-1*
drop_diskgroup_file_clause, *5-1*
drop_filegroup_clause, *5-1*
drop_index_partition, *5-1*
drop_logfile_clauses, *5-1*
drop_period_clause, *5-1*
drop_table_partition, *5-1*
drop_table_subpartition, *5-1*
ds_iso_format of TO_DSINTERVAL function, *5-1*
DUMP function, *2-1*

E

EDIT SQL*Plus command, *A-3*
else_clause, *5-1*
EMPTY_BLOB function, *2-1*
EMPTY_CLOB function, *2-1*
enable_disable_clause, *5-1*
enable_disable_volume, *5-1*
enable_pluggable_database, *5-1*
encryption_spec, *5-1*
end_session_clauses, *5-1*
EQUALS_PATH condition, *4-1*
error_logging_clause, *5-1*
evaluation_edition_clause, *5-1*
exceptions_clause, *5-1*
exchange_partition_subpart, *5-1*
EXECUTE SQL*Plus command, *A-4*
EXISTS condition, *4-1*
EXISTSNODE function, *2-1*
EXIT SQL*Plus command, *A-4*
EXP function, *2-1*
EXPLAIN PLAN statement, *1-1*
export_keys, *5-1*
expr, *5-1*
expression_list, *5-1*
expressions, *3-1*
  see also SQL expressions, *3-1*
extended_attribute_clause, *5-1*
extent_management_clause, *5-1*
external_part_subpart_data_props, *5-1*
external_table_clause, *5-1*
external_table_data_props, *5-1*
EXTRACT (datetime) function, *2-1*
EXTRACT (XML) function, *2-1*
EXTRACTVALUE function, *2-1*

F

failover_clause, *5-1*
FEATURE_COMPARE function, *2-1*
FEATURE_DETAILS (analytic) function, *2-1*
FEATURE_DETAILS function, *2-1*
FEATURE_ID (analytic) function, *2-1*
FEATURE_ID function, *2-1*
FEATURE_SET (analytic) function, *2-1*
FEATURE_SET function, *2-1*
FEATURE_VALUE (analytic) function, *2-1*
FEATURE_VALUE function, *2-1*
file_name_convert, *5-1*
file_owner_clause, *5-1*
file_permissions_clause, *5-1*
file_specification, *5-1*
filegroup_clauses, *5-1*
filter_condition, *5-1*
FIRST function, *2-1*
FIRST_VALUE function, *2-1*
FLASHBACK DATABASE statement, *1-1*
FLASHBACK TABLE statement, *1-1*
flashback_archive_clause, *5-1*
flashback_archive_quota, *5-1*
flashback_archive_retention, *5-1*
flashback_mode_clause, *5-1*
flashback_query_clause, *5-1*
floating-point conditions, *4-1*
FLOOR function, *2-1*
following_boundary, *5-1*
for_refresh_clause, *5-1*
for_update_clause, *5-1*
format models, *7-1*
  date format models, *7-3*
  number format models, *7-1*
FROM_TZ function, *2-1*
full_database_recovery, *5-1*

fully_qualified_file_name, *5-1*
function expressions, *3-1*
function_association, *5-1*
functions, *2-1*
    see also SQL functions, *2-1*

G

general_recovery, *5-1*
GET SQL*Plus command, *A-3*
global_partitioned_index, *5-1*
GRANT statement, *1-1*
grant_object_privileges, *5-1*
grant_roles_to_programs, *5-1*
grant_system_privileges, *5-1*
grantee_clause, *5-1*
grantee_identified_by, *5-1*
GRAPHIC data type
    DB2, *6-6*
    SQL/DS, *6-6*
GREATEST function, *2-1*
group comparison conditions, *4-1*
group separator
    specifying, *7-2*
group_by_clause, *5-1*
GROUP_ID function, *2-1*
GROUPING function, *2-1*
grouping_expression_list, *5-1*
GROUPING_ID function, *2-1*
grouping_sets_clause, *5-1*

H

hash_partitions, *5-1*
hash_partitions_by_quantity, *5-1*
hash_subparts_by_quantity, *5-1*
heap_org_table_clause, *5-1*
HELP SQL*Plus command, *A-1*
hexadecimal value
    returning, *7-3*
HEXTORAW function, *2-1*
hier_ancestor_expression, *5-1*
hier_attr_clause, *5-1*
hier_attr_name, *5-1*
hier_attrs_clause, *5-1*
hier_lead_lag_clause, *5-1*
hier_lead_lag_expression, *5-1*
hier_navigation_expression, *5-1*
hier_parent_expression, *5-1*
hier_ref, *5-1*
hier_using_clause, *5-1*
hierarchical_query_clause, *5-1*
hierarchy_clause, *5-1*
hierarchy_ref, *5-1*
HOST SQL*Plus command, *A-1*

I

identity_clause, *5-1*
identity_options, *5-1*
ilm_clause, *5-1*
ilm_compression_policy, *5-1*
ilm_inmemory_policy, *5-1*
ilm_policy_clause, *5-1*
ilm_tiering_policy, *5-1*
ilm_time_period, *5-1*
implementation_clause, *5-1*
import_keys, *5-1*
IN condition, *4-1*
incomplete_file_name, *5-1*
index_attributes, *5-1*
index_compression, *5-1*
index_expr, *5-1*
index_org_overflow_clause, *5-1*
index_org_table_clause, *5-1*
index_partition_description, *5-1*
index_partitioning_clause, *5-1*
index_properties, *5-1*
index_subpartition_clause, *5-1*
indexing_clause, *5-1*
individual_hash_partitions, *5-1*
individual_hash_subparts, *5-1*
INITCAP function, *2-1*
inline_constraint, *5-1*
inline_ref_constraint, *5-1*
inmemory_attributes, *5-1*
inmemory_clause, *5-1*
inmemory_column_clause, *5-1*
inmemory_distribute, *5-1*
inmemory_duplicate, *5-1*
inmemory_memcompress, *5-1*
inmemory_priority, *5-1*
inmemory_table_clause, *5-1*
inner_cross_join_clause, *5-1*
INPUT SQL*Plus command, *A-3*
INSERT statement, *1-1*
insert_into_clause, *5-1*
instance_clauses, *5-1*
instances_clause, *5-1*
INSTR function, *2-1*
integer, *5-1*
INTERVAL expressions, *3-1*
interval_day_to_second, *5-1*
interval_year_to_month, *5-1*
into_clause, *5-1*
invoker_rights_clause, *5-1*
IS A SET condition, *4-1*
IS ANY condition, *4-1*
IS EMPTY condition, *4-1*
IS JSON condition, *4-1*
IS OF *type* condition, *4-1*
IS PRESENT condition, *4-1*

ITERATION_NUMBER function, *2-1*

J

join_clause, *5-1*
JSON object access expressions, *3-1*
JSON_agg_returning_clause, *5-1*
JSON_ARRAY function, *2-1*
JSON_ARRAYAGG function, *2-1*
JSON_column_definition, *5-1*
JSON_columns_clause, *5-1*
JSON_DATAGUIDE function, *2-1*
JSON_EXISTS condition, *4-1*
JSON_exists_column, *5-1*
JSON_exists_on_error_clause, *5-1*
JSON_nested_path, *5-1*
JSON_OBJECT function, *2-1*
JSON_OBJECTAGG function, *2-1*
JSON_on_null_clause, *5-1*
JSON_passing_clause, *5-1*
JSON_QUERY function, *2-1*
JSON_query_column, *5-1*
JSON_query_on_empty_clause, *5-1*
JSON_query_on_error_clause, *5-1*
JSON_query_return_type, *5-1*
JSON_query_returning_clause, *5-1*
JSON_query_wrapper_clause, *5-1*
JSON_returning_clause, *5-1*
JSON_TABLE function, *2-1*
JSON_table_on_error_clause, *5-1*
JSON_TEXTCONTAINS condition, *4-1*
JSON_TRANSFORM function, *2-1*
JSON_VALUE function, *2-1*
JSON_value_column, *5-1*
JSON_value_on_empty_clause, *5-1*
JSON_value_on_error_clause, *5-1*
JSON_value_return_type, *5-1*
JSON_value_returning_clause, *5-1*

K

key_clause, *5-1*
key_management_clauses, *5-1*
keystore_clause, *5-1*
keystore_management_clauses, *5-1*

L

LAG function, *2-1*
large_object_datatypes, *6-2*
LAST function, *2-1*
LAST_DAY function, *2-1*
LAST_VALUE function, *2-1*
LEAD function, *2-1*
lead_lag_clause, *5-1*
lead_lag_expression, *5-1*
lead_lag_function_name, *5-1*
LEAST function, *2-1*
LENGTH function, *2-1*
level_clause, *5-1*
level_hier_clause, *5-1*
level_member_literal, *5-1*
level_specification, *5-1*
levels_clause, *5-1*
LIKE condition, *4-1*
LIST SQL*Plus command, *A-3*
list_partition_desc, *5-1*
list_partitions, *5-1*
list_partitionset_clause, *5-1*
list_partitionset_desc, *5-1*
list_subpartition_desc, *5-1*
list_values, *5-1*
list_values_clause, *5-1*
LISTAGG function, *2-1*
listagg_overflow_clause, *5-1*
LN function, *2-1*
LNNVL function, *2-1*
LOB_compression_clause, *5-1*
LOB_deduplicate_clause, *5-1*
LOB_parameters, *5-1*
LOB_partition_storage, *5-1*
LOB_partitioning_storage, *5-1*
LOB_retention_storage, *5-1*
LOB_storage_clause, *5-1*
LOB_storage_parameters, *5-1*
local_domain_index_clause, *5-1*
local_partitioned_index, *5-1*
local_XMLIndex_clause, *5-1*
locale independent, *7-4*
LOCALTIMESTAMP function, *2-1*
LOCK TABLE statement, *1-1*
lockdown_features, *5-1*
lockdown_options, *5-1*
lockdown_statements, *5-1*
LOG function, *2-1*
logfile_clause, *5-1*
logfile_clauses, *5-1*
logfile_descriptor, *5-1*
logging_clause, *5-1*
logical conditions, *4-1*
LONG VARGRAPHIC data type
  DB2, *6-6*
  SQL/DS, *6-6*
long_and_raw_datatypes, *6-2*
LOWER function, *2-1*
LPAD function, *2-1*
LTRIM function, *2-1*

M

main_model, *5-1*
MAKE_REF function, *2-1*

managed_standby_recovery, *5-1*
mapping_table_clauses, *5-1*
materialized_view_props, *5-1*
MAX function, *2-1*
maximize_standby_db_clause, *5-1*
maxsize_clause, *5-1*
meas_aggregate_clause, *5-1*
measure, *5-1*
measure_ref, *5-1*
measures_clause, *5-1*
media_types, *6-5*
MEDIAN function, *2-1*
MEMBER condition, *4-1*
member_expression, *5-1*
MERGE statement, *1-1*
merge_insert_clause, *5-1*
merge_into_existing_keystore, *5-1*
merge_into_new_keystore, *5-1*
merge_table_partitions, *5-1*
merge_table_subpartitions, *5-1*
merge_update_clause, *5-1*
migrate_key, *5-1*
MIN function, *2-1*
mining_analytic_clause, *5-1*
mining_attribute_clause, *5-1*
MOD function, *2-1*
model expressions, *3-1*
model_clause, *5-1*
model_column_clauses, *5-1*
model_iterate_clause, *5-1*
model_rules_clause, *5-1*
modify_col_properties, *5-1*
modify_col_substitutable, *5-1*
modify_col_visibility, *5-1*
modify_collection_retrieval, *5-1*
modify_column_clauses, *5-1*
modify_diskgroup_file, *5-1*
modify_filegroup_clause, *5-1*
modify_hash_partition, *5-1*
modify_index_default_attrs, *5-1*
modify_index_partition, *5-1*
modify_index_subpartition, *5-1*
modify_list_partition, *5-1*
modify_LOB_parameters, *5-1*
modify_LOB_storage_clause, *5-1*
modify_mv_column_clause, *5-1*
modify_opaque_type, *5-1*
modify_range_partition, *5-1*
modify_table_default_attrs, *5-1*
modify_table_partition, *5-1*
modify_table_subpartition, *5-1*
modify_to_partitioned, *5-1*
modify_virtcol_properties, *5-1*
modify_volume_clause, *5-1*
MONTHS_BETWEEN function, *2-1*
move_datafile_clause, *5-1*
move_mv_log_clause, *5-1*
move_table_clause, *5-1*
move_table_partition, *5-1*
move_table_subpartition, *5-1*
move_to_filegroup_clause, *5-1*
multi_column_for_loop, *5-1*
multi_table_insert, *5-1*
multiset_except, *5-1*
multiset_intersect, *5-1*
multiset_union, *5-1*
mv_log_augmentation, *5-1*
mv_log_purge_clause, *5-1*

N

named_member_keys, *5-1*
NANVL function, *2-1*
NCHR function, *2-1*
nested_table_col_properties, *5-1*
nested_table_partition_spec, *5-1*
NEW_TIME function, *2-1*
new_values_clause, *5-1*
NEXT_DAY function, *2-1*
NLS_CHARSET_DECL_LEN function, *2-1*
NLS_CHARSET_ID function, *2-1*
NLS_CHARSET_NAME function, *2-1*
NLS_COLLATION_ID function, *2-1*
NLS_COLLATION_NAME function, *2-1*
NLS_INITCAP function, *2-1*
NLS_LOWER function, *2-1*
NLS_UPPER function, *2-1*
NLSSORT function, *2-1*
NOAUDIT (Traditional Auditing) statement, *1-1*
NOAUDIT (Unified Auditing) statement, *1-1*
NTH_VALUE function, *2-1*
NTILE function, *2-1*
null conditions, *4-1*
NULLIF function, *2-1*
number, *5-1*
number format elements, *7-1*
number format models, *7-1*
number_datatypes, *6-2*
numeric_file_name, *5-1*
NUMTODSINTERVAL function, *2-1*
NUMTOYMINTERVAL function, *2-1*
NVL function, *2-1*
NVL2 function, *2-1*

O

object access expressions, *3-1*
object_properties, *5-1*
object_step, *5-1*
object_table, *5-1*
object_table_substitution, *5-1*
object_type_col_properties, *5-1*

object_view_clause, *5-1*
OID_clause, *5-1*
OID_index_clause, *5-1*
on_comp_partitioned_table, *5-1*
on_hash_partitioned_table, *5-1*
on_list_partitioned_table, *5-1*
on_object_clause, *5-1*
on_range_partitioned_table, *5-1*
open_keystore, *5-1*
option_values, *5-1*
ORA_DM_PARTITION_NAME function, *2-1*
ORA_DST_AFFECTED function, *2-1*
ORA_DST_CONVERT function, *2-1*
ORA_DST_ERROR function, *2-1*
ORA_HASH function, *2-1*
ORA_INVOKING_USER function, *2-1*
ORA_INVOKING_USERID function, *2-1*
Oracle built-in data types, *6-1*, *6-2*
Oracle-supplied data types, *6-1*, *6-5*
order_by_clause, *5-1*
ordinality_column, *5-1*
out_of_line_constraint, *5-1*
out_of_line_part_storage, *5-1*
out_of_line_ref_constraint, *5-1*
outer_join_clause, *5-1*
outer_join_type, *5-1*

P

parallel_clause, *5-1*
parallel_pdb_creation_clause, *5-1*
partial_database_recovery, *5-1*
partial_index_clause, *5-1*
partition_attributes, *5-1*
partition_extended_name, *5-1*
partition_extended_names, *5-1*
partition_extension_clause, *5-1*
partition_or_key_value, *5-1*
partition_spec, *5-1*
partitioning_storage_clause, *5-1*
partitionset_clauses, *5-1*
password_parameters, *5-1*
PATH function, *2-1*
path_prefix_clause, *5-1*
pdb_change_state, *5-1*
pdb_change_state_from_root, *5-1*
pdb_close, *5-1*
pdb_datafile_clause, *5-1*
pdb_dba_roles, *5-1*
pdb_force_logging_clause, *5-1*
pdb_general_recovery, *5-1*
pdb_logging_clauses, *5-1*
pdb_open, *5-1*
pdb_recovery_clauses, *5-1*
pdb_refresh_mode_clause, *5-1*
pdb_save_or_discard_state, *5-1*
pdb_settings_clauses, *5-1*
pdb_storage_clause, *5-1*
pdb_unplug_clause, *5-1*
PERCENT_RANK (aggregate) function, *2-1*
PERCENT_RANK (analytic) function, *2-1*
PERCENTILE_CONT function, *2-1*
PERCENTILE_DISC function, *2-1*
period_definition, *5-1*
permanent_tablespace_attrs, *5-1*
permanent_tablespace_clause, *5-1*
physical_attributes_clause, *5-1*
physical_properties, *5-1*
pivot_clause, *5-1*
pivot_for_clause, *5-1*
pivot_in_clause, *5-1*
placeholder expressions, *3-1*
plsql_declarations, *5-1*
pos_member_keys, *5-1*
POWER function, *2-1*
POWERMULTISET function, *2-1*
POWERMULTISET_BY_CARDINALITY function, *2-1*
preceding_boundary, *5-1*
PREDICTION (analytic) function, *2-1*
PREDICTION function, *2-1*
PREDICTION_BOUNDS function, *2-1*
PREDICTION_COST (analytic) function, *2-1*
PREDICTION_COST function, *2-1*
PREDICTION_DETAILS (analytic) function, *2-1*
PREDICTION_DETAILS function, *2-1*
PREDICTION_PROBABILITY (analytic) function, *2-1*
PREDICTION_PROBABILITY function, *2-1*
PREDICTION_SET (analytic) function, *2-1*
PREDICTION_SET function, *2-1*
prefix_compression, *5-1*
PRESENTNNV function, *2-1*
PRESENTV function, *2-1*
PREVIOUS function, *2-1*
privilege_audit_clause, *5-1*
program_unit, *5-1*
proxy_clause, *5-1*
PURGE statement, *1-1*

Q

qdr_expression, *5-1*
qualified_disk_clause, *5-1*
qualified_template_clause, *5-1*
qualifier, *5-1*
query_block, *5-1*
query_partition_clause, *5-1*
query_rewrite_clause, *5-1*
query_table_expression, *5-1*
quiesce_clauses, *5-1*
QUIT SQL*Plus command, *A-4*

quotagroup_clauses, *5-1*

R

range_partition_desc, *5-1*
range_partitions, *5-1*
range_partitionset_clause, *5-1*
range_partitionset_desc, *5-1*
range_subpartition_desc, *5-1*
range_values_clause, *5-1*
RANK (aggregate) function, *2-1*
RANK (analytic) function, *2-1*
RATIO_TO_REPORT function, *2-1*
RAWTOHEX function, *2-1*
RAWTONHEX function, *2-1*
read_only_clause, *5-1*
rebalance_diskgroup_clause, *5-1*
rebuild_clause, *5-1*
records_per_block_clause, *5-1*
recovery_clauses, *5-1*
redo_log_file_spec, *5-1*
redo_thread_clauses
    see instance_clauses, *5-1*
redundancy_clause, *5-1*
REF function, *2-1*
reference_model, *5-1*
reference_partition_desc, *5-1*
reference_partitioning, *5-1*
references_clause, *5-1*
REFTOHEX function, *2-1*
REGEXP_COUNT function, *2-1*
REGEXP_INSTR function, *2-1*
REGEXP_LIKE condition, *4-1*
REGEXP_REPLACE function, *2-1*
REGEXP_SUBSTR function, *2-1*
register_logfile_clause, *5-1*
REGR_AVGX function, *2-1*
REGR_AVGY function, *2-1*
REGR_COUNT function, *2-1*
REGR_INTERCEPT function, *2-1*
REGR_R2 function, *2-1*
REGR_SLOPE function, *2-1*
REGR_SXX function, *2-1*
REGR_SXY function, *2-1*
REGR_SYY function, *2-1*
relational_properties, *5-1*
relational_table, *5-1*
relocate_clause, *5-1*
REMAINDER function, *2-1*
RENAME statement, *1-1*
rename_column_clause, *5-1*
rename_disk_clause, *5-1*
rename_index_partition, *5-1*
rename_partition_subpart, *5-1*
REPLACE function, *2-1*
replace_disk_clause, *5-1*
resize_disk_clause, *5-1*
resource_parameters, *5-1*
return_rows_clause, *5-1*
returning_clause, *5-1*
reverse_migrate_key, *5-1*
REVOKE statement, *1-1*
revoke_object_privileges, *5-1*
revoke_roles_from_programs, *5-1*
revoke_system_privileges, *5-1*
revokee_clause, *5-1*
role_audit_clause, *5-1*
ROLLBACK statement, *1-1*
rolling_migration_clauses, *5-1*
rolling_patch_clauses, *5-1*
rollup_cube_clause, *5-1*
ROUND (date) function, *2-1*
ROUND (number) function, *2-1*
routine_clause, *5-1*
row_limiting_clause, *5-1*
row_movement_clause, *5-1*
ROW_NUMBER function, *2-1*
row_pattern, *5-1*
row_pattern_aggregate_func, *5-1*
row_pattern_classifier_func, *5-1*
row_pattern_clause, *5-1*
row_pattern_definition, *5-1*
row_pattern_definition_list, *5-1*
row_pattern_factor, *5-1*
row_pattern_match_num_func, *5-1*
row_pattern_measure_column, *5-1*
row_pattern_measures, *5-1*
row_pattern_nav_compound, *5-1*
row_pattern_nav_logical, *5-1*
row_pattern_nav_physical, *5-1*
row_pattern_navigation_func, *5-1*
row_pattern_order_by, *5-1*
row_pattern_partition_by, *5-1*
row_pattern_permute, *5-1*
row_pattern_primary, *5-1*
row_pattern_quantifier, *5-1*
row_pattern_rec_func, *5-1*
row_pattern_rows_per_match, *5-1*
row_pattern_skip_to, *5-1*
row_pattern_subset_clause, *5-1*
row_pattern_subset_item, *5-1*
row_pattern_term, *5-1*
rowid_datatypes, *6-2*
ROWIDTOCHAR function, *2-1*
ROWTONCHAR function, *2-1*
RPAD function, *2-1*
RTRIM function, *2-1*
RUN SQL*Plus command, *A-4*

S

sample_clause, *5-1*

SAVE SQL*Plus command, *A-3*
SAVEPOINT statement, *1-1*
scalar subquery expressions, *3-1*
scientific notation, *7-2*
SCN_TO_TIMESTAMP function, *2-1*
scoped_table_ref_constraint, *5-1*
scrub_clause, *5-1*
search_clause, *5-1*
searched_case_expression, *5-1*
secret_management_clauses, *5-1*
security_clause, *5-1*
security_clauses, *5-1*
segment_attributes_clause, *5-1*
segment_management_clause, *5-1*
SELECT statement, *1-1*
select_list, *5-1*
service_name_convert, *5-1*
SESSIONTIMEZONE function, *2-1*
SET CONSTRAINT statement, *1-1*
SET function, *2-1*
SET ROLE statement, *1-1*
SET SQL*Plus command, *A-2*
SET TRANSACTION statement, *1-1*
set_encryption_key, *5-1*
set_key, *5-1*
set_key_tag, *5-1*
set_parameter_clause, *5-1*
set_subpartition_template, *5-1*
set_time_zone_clause, *5-1*
share_clause, *5-1*
share_of_expression, *5-1*
sharing_clause, *5-1*
SHOW SQL*Plus command, *A-2*
shrink_clause, *5-1*
SHUTDOWN SQL*Plus command, *A-4*
shutdown_dispatcher_clause, *5-1*
SIGN function, *2-1*
simple comparison conditions, *4-1*
simple expressions, *3-1*
simple_case_expression, *5-1*
SIN function, *2-1*
single_column_for_loop, *5-1*
single_table_insert, *5-1*
SINH function, *2-1*
size_clause, *5-1*
SOUNDEX function, *2-1*
source_file_directory, *5-1*
source_file_name_convert, *5-1*
spatial_types, *6-5*
split_index_partition, *5-1*
split_nested_table_part, *5-1*
split_table_partition, *5-1*
split_table_subpartition, *5-1*
SPOOL SQL*Plus command, *A-3*
SQL conditions, *4-1*
  BETWEEN condition, *4-1*
SQL conditions *(continued)*
  compound conditions, *4-1*
  EQUALS_PATH condition, *4-1*
  EXISTS condition, *4-1*
  floating-point conditions, *4-1*
  group comparison conditions, *4-1*
  IN condition, *4-1*
  IS A SET condition, *4-1*
  IS ANY condition, *4-1*
  IS EMPTY condition, *4-1*
  IS JSON condition, *4-1*
  IS OF *type* condition, *4-1*
  IS PRESENT condition, *4-1*
  JSON_EXISTS condition, *4-1*
  JSON_TEXTCONTAINS condition, *4-1*
  LIKE condition, *4-1*
  logical conditions, *4-1*
  MEMBER condition, *4-1*
  null conditions, *4-1*
  REGEXP_LIKE condition, *4-1*
  simple comparison conditions, *4-1*
  SUBMULTISET condition, *4-1*
  UNDER_PATH condition, *4-1*
SQL expressions, *3-1*
  calculated measure expressions, *3-1*
  CASE expressions, *3-1*
  column expressions, *3-1*
  compound expressions, *3-1*
  CURSOR expressions, *3-1*
  datetime expressions, *3-1*
  function expressions, *3-1*
  INTERVAL expressions, *3-1*
  JSON object access expressions, *3-1*
  model expressions, *3-1*
  object access expressions, *3-1*
  placeholder expressions, *3-1*
  scalar subquery expressions, *3-1*
  simple expressions, *3-1*
  type constructor expressions, *3-1*
SQL functions, *2-1*
  ABS, *2-1*
  ACOS, *2-1*
  ADD_MONTHS, *2-1*
  aggregate functions, *2-1*
  analytic functions, *2-1*
  APPROX_COUNT_DISTINCT, *2-1*
  APPROX_COUNT_DISTINCT_AGG, *2-1*
  APPROX_COUNT_DISTINCT_DETAIL, *2-1*
  APPROX_MEDIAN, *2-1*
  APPROX_PERCENTILE, *2-1*
  APPROX_PERCENTILE_AGG, *2-1*
  APPROX_PERCENTILE_DETAIL, *2-1*
  ASCII, *2-1*
  ASCIISTR, *2-1*
  ASIN, *2-1*
  ATAN, *2-1*

SQL functions *(continued)*
ATAN2, *2-1*
AVG, *2-1*
BFILENAME, *2-1*
BIN_TO_NUM, *2-1*
BITAND, *2-1*
CARDINALITY, *2-1*
CAST, *2-1*
CEIL, *2-1*
CHARTOROWID, *2-1*
CHR, *2-1*
CLUSTER_DETAILS, *2-1*
CLUSTER_DETAILS (analytic), *2-1*
CLUSTER_DISTANCE, *2-1*
CLUSTER_DISTANCE (analytic), *2-1*
CLUSTER_ID, *2-1*
CLUSTER_ID (analytic), *2-1*
CLUSTER_PROBABILITY, *2-1*
CLUSTER_PROBABILITY (analytic), *2-1*
CLUSTER_SET, *2-1*
CLUSTER_SET (analytic), *2-1*
COALESCE, *2-1*
COLLATION, *2-1*
COLLECT, *2-1*
COMPOSE, *2-1*
CON_DBID_TO_ID, *2-1*
CON_GUID_TO_ID, *2-1*
CON_NAME_TO_ID, *2-1*
CON_UID_TO_ID, *2-1*
CONCAT, *2-1*
CONVERT, *2-1*
CORR, *2-1*
CORR_K, *2-1*
CORR_S, *2-1*
COS, *2-1*
COSH, *2-1*
COUNT, *2-1*
COVAR_POP, *2-1*
COVAR_SAMP, *2-1*
CUBE_TABLE, *2-1*
CUME_DIST (aggregate), *2-1*
CUME_DIST (analytic), *2-1*
CURRENT_DATE, *2-1*
CURRENT_TIMESTAMP, *2-1*
CV, *2-1*
DATAOBJ_TO_MAT_PARTITION, *2-1*
DATAOBJ_TO_PARTITION, *2-1*
DBTIMEZONE, *2-1*
DECODE, *2-1*
DECOMPOSE, *2-1*
DENSE_RANK (aggregate), *2-1*
DENSE_RANK (analytic), *2-1*
DEPTH, *2-1*
DEREF, *2-1*
DUMP, *2-1*
EMPTY_BLOB, *2-1*

SQL functions *(continued)*
EMPTY_CLOB, *2-1*
EXISTSNODE, *2-1*
EXP, *2-1*
EXTRACT (datetime), *2-1*
EXTRACT (XML), *2-1*
EXTRACTVALUE, *2-1*
FEATURE_COMPARE, *2-1*
FEATURE_DETAILS, *2-1*
FEATURE_DETAILS (analytic), *2-1*
FEATURE_ID, *2-1*
FEATURE_ID (analytic), *2-1*
FEATURE_SET, *2-1*
FEATURE_SET (analytic), *2-1*
FEATURE_VALUE, *2-1*
FEATURE_VALUE (analytic), *2-1*
FIRST, *2-1*
FIRST_VALUE, *2-1*
FLOOR, *2-1*
FROM_TZ, *2-1*
GREATEST, *2-1*
GROUP_ID, *2-1*
GROUPING, *2-1*
GROUPING_ID, *2-1*
HEXTORAW, *2-1*
INITCAP, *2-1*
INSTR, *2-1*
ITERATION_NUMBER, *2-1*
JSON_ARRAY, *2-1*
JSON_ARRAYAGG, *2-1*
JSON_DATAGUIDE, *2-1*
JSON_OBJECT, *2-1*
JSON_OBJECTAGG, *2-1*
JSON_QUERY, *2-1*
JSON_TABLE, *2-1*
JSON_TRANSFORM, *2-1*
JSON_VALUE, *2-1*
LAG, *2-1*
LAST, *2-1*
LAST_DAY, *2-1*
LAST_VALUE, *2-1*
LEAD, *2-1*
LEAST, *2-1*
LENGTH, *2-1*
LISTAGG, *2-1*
LN, *2-1*
LNNVL, *2-1*
LOCALTIMESTAMP, *2-1*
LOG, *2-1*
LOWER, *2-1*
LPAD, *2-1*
LTRIM, *2-1*
MAKE_REF, *2-1*
MAX, *2-1*
MEDIAN, *2-1*
MIN, *2-1*

SQL functions *(continued)*
MOD, *2-1*
MONTHS_BETWEEN, *2-1*
NANVL, *2-1*
NCGR, *2-1*
NEW_TIME, *2-1*
NEXT_DAY, *2-1*
NLS_CHARSET_DECL_LEN, *2-1*
NLS_CHARSET_ID, *2-1*
NLS_CHARSET_NAME, *2-1*
NLS_COLLATION_ID, *2-1*
NLS_COLLATION_NAME, *2-1*
NLS_INITCAP, *2-1*
NLS_LOWER, *2-1*
NLS_UPPER, *2-1*
NLSSORT, *2-1*
NTH_VALUE, *2-1*
NTILE, *2-1*
NULLIF, *2-1*
NUMTODSINTERVAL, *2-1*
NUMTOYMINTERVAL, *2-1*
NVL, *2-1*
NVL2, *2-1*
ORA_DM_PARTITION_NAME, *2-1*
ORA_DST_AFFECTED, *2-1*
ORA_DST_CONVERT, *2-1*
ORA_DST_ERROR, *2-1*
ORA_HASH, *2-1*
ORA_INVOKING_USER, *2-1*
ORA_INVOKING_USERID, *2-1*
PATH, *2-1*
PERCENT_RANK (aggregate), *2-1*
PERCENT_RANK (analytic), *2-1*
PERCENTILE_CONT, *2-1*
PERCENTILE_DISC, *2-1*
POWER, *2-1*
POWERMULTISET, *2-1*
POWERMULTISET_BY_CARDINALITY, *2-1*
PREDICTION, *2-1*
PREDICTION (analytic), *2-1*
PREDICTION_BOUNDS, *2-1*
PREDICTION_COST, *2-1*
PREDICTION_COST (analytic), *2-1*
PREDICTION_DETAILS, *2-1*
PREDICTION_DETAILS (analytic), *2-1*
PREDICTION_PROBABILITY, *2-1*
PREDICTION_PROBABILITY (analytic), *2-1*
PREDICTION_SET, *2-1*
PREDICTION_SET (analytic), *2-1*
PRESENTNNV, *2-1*
PRESENTV, *2-1*
PREVIOUS, *2-1*
RANK (aggregate), *2-1*
RANK (analytic), *2-1*

SQL functions *(continued)*
RATIO_TO_REPORT, *2-1*
RAWTOHEX, *2-1*
RAWTONHEX, *2-1*
REF, *2-1*
REFTOHEX, *2-1*
REGEXP_COUNT, *2-1*
REGEXP_INSTR, *2-1*
REGEXP_REPLACE, *2-1*
REGEXP_SUBSTR, *2-1*
REGR_AVGX, *2-1*
REGR_AVGY, *2-1*
REGR_COUNT, *2-1*
REGR_INTERCEPT, *2-1*
REGR_R2, *2-1*
REGR_SLOPE, *2-1*
REGR_SXX, *2-1*
REGR_SXY, *2-1*
REGR_SYY, *2-1*
REMAINDER, *2-1*
REPLACE, *2-1*
ROUND (date), *2-1*
ROUND (number), *2-1*
ROW_NUMBER, *2-1*
ROWIDTOCHAR, *2-1*
ROWTONCHAR, *2-1*
RPAD, *2-1*
RTRIM, *2-1*
SCN_TO_TIMESTAMP, *2-1*
SESSIONTIMEZONE, *2-1*
SET, *2-1*
SIGN, *2-1*
SIN, *2-1*
SINH, *2-1*
SOUNDEX, *2-1*
SQRT, *2-1*
STANDARD_HASH, *2-1*
STATS_BINOMIAL_TEST, *2-1*
STATS_CROSSTAB, *2-1*
STATS_F_TEST, *2-1*
STATS_KS_TEST, *2-1*
STATS_MODE, *2-1*
STATS_MW_TEST, *2-1*
STATS_ONE_WAY_ANOVA, *2-1*
STATS_T_TEST_INDEP, *2-1*
STATS_T_TEST_INDEPU, *2-1*
STATS_T_TEST_ONE, *2-1*
STATS_T_TEST_PAIRED, *2-1*
STATS_WSR_TEST, *2-1*
STDDEV, *2-1*
STDDEV_POP, *2-1*
STDDEV_SAMP, *2-1*
SUBSTR, *2-1*
SUM, *2-1*
SYS_CONNECT_BY_PATH, *2-1*
SYS_CONTEXT, *2-1*

SQL functions *(continued)*
SYS_DBURIGEN, *2-1*
SYS_EXTRACT_UTC, *2-1*
SYS_GUID, *2-1*
SYS_OP_ZONE_ID, *2-1*
SYS_TYPEID, *2-1*
SYS_XMLAGG, *2-1*
SYS_XMLGEN, *2-1*
SYSDATE, *2-1*
SYSTIMESTAMP, *2-1*
TAN, *2-1*
TANH, *2-1*
TIMESTAMP_TO_SCN, *2-1*
TO_APPROX_COUNT_DISTINCT, *2-1*
TO_APPROX_PERCENTILE, *2-1*
TO_BINARY_DOUBLE, *2-1*
TO_BINARY_FLOAT, *2-1*
TO_BLOB (bfile), *2-1*
TO_BLOB (raw), *2-1*
TO_CHAR (bfile|blob), *2-1*
TO_CHAR (character), *2-1*
TO_CHAR (datetime), *2-1*
TO_CHAR (number), *2-1*
TO_CLOB (bfile|blob), *2-1*
TO_CLOB (character), *2-1*
TO_DATE, *2-1*
TO_DSINTERVAL, *2-1*
TO_LOB, *2-1*
TO_MULTI_BYTE, *2-1*
TO_NCHAR (character), *2-1*
TO_NCHAR (datetime), *2-1*
TO_NCHAR (number), *2-1*
TO_NCLOB, *2-1*
TO_NUMBER, *2-1*
TO_SINGLE_BYTE, *2-1*
TO_TIMESTAMP, *2-1*
TO_TIMESTAMP_TZ, *2-1*
TO_YMINTERVAL, *2-1*
TRANSLATE, *2-1*
TRANSLATE...USING, *2-1*
TREAT, *2-1*
TRIM, *2-1*
TRUNC (date), *2-1*
TRUNC (number), *2-1*
TZ_OFFSET, *2-1*
UID, *2-1*
UNISTR, *2-1*
UPPER, *2-1*
USER, *2-1*
user-defined functions, *2-1*
USERENV, *2-1*
VALIDATE_CONVERSION, *2-1*
VALUE, *2-1*
VAR_POP, *2-1*
VAR_SAMP, *2-1*
VARIANCE, *2-1*
SQL functions *(continued)*
VSIZE, *2-1*
WIDTH_BUCKET, *2-1*
XMLAGG, *2-1*
XMLCAST, *2-1*
XMLCDATA, *2-1*
XMLCOLATTVAL, *2-1*
XMLCOMMENT, *2-1*
XMLCONCAT, *2-1*
XMLDIFF, *2-1*
XMLELEMENT, *2-1*
XMLEXISTS, *2-1*
XMLFOREST, *2-1*
XMLISVALID, *2-1*
XMLPARSE, *2-1*
XMLPATCH, *2-1*
XMLPI, *2-1*
XMLQUERY, *2-1*
XMLROOT, *2-1*
XMLSEQUENCE, *2-1*
XMLSERIALIZE, *2-1*
XMLTABLE, *2-1*
XMLTRANSFORM, *2-1*
SQL statements, *1-1*
ADMINISTER KEY MANAGEMENT, *1-1*
ALTER ANALYTIC VIEW, *1-1*
ALTER ATTRIBUTE DIMENSION, *1-1*
ALTER AUDIT POLICY, *1-1*
ALTER CLUSTER, *1-1*
ALTER DATABASE, *1-1*
ALTER DATABASE LINK, *1-1*
ALTER DIMENSION, *1-1*
ALTER DISKGROUP, *1-1*
ALTER FLASHBACK ARCHIVE, *1-1*
ALTER FUNCTION, *1-1*
ALTER HIERARCHY, *1-1*
ALTER INDEX, *1-1*
ALTER INDEXTYPE, *1-1*
ALTER INMEMORY JOIN GROUP, *1-1*
ALTER JAVA, *1-1*
ALTER LIBRARY, *1-1*
ALTER LOCKDOWN PROFILE, *1-1*
ALTER MATERIALIZED VIEW, *1-1*
ALTER MATERIALIZED VIEW LOG, *1-1*
ALTER MATERIALIZED ZONEMAP, *1-1*
ALTER OPERATOR, *1-1*
ALTER OUTLINE, *1-1*
ALTER PACKAGE, *1-1*
ALTER PLUGGABLE DATABASE, *1-1*
ALTER PROCEDURE, *1-1*
ALTER PROFILE, *1-1*
ALTER PROPERTY GRAPH, *1-1*
ALTER RESOURCE COST, *1-1*
ALTER ROLE, *1-1*
ALTER ROLLBACK SEGMENT, *1-1*
ALTER SEQUENCE, *1-1*

SQL statements *(continued)*
ALTER SESSION, *1-1*
ALTER SYNONYM, *1-1*
ALTER SYSTEM, *1-1*
ALTER TABLE, *1-1*
ALTER TABLESPACE, *1-1*
ALTER TABLESPACE SET, *1-1*
ALTER TRIGGER, *1-1*
ALTER TYPE, *1-1*
ALTER USER, *1-1*
ALTER VIEW, *1-1*
ANALYZE, *1-1*
ASSOCIATE STATISTICS, *1-1*
AUDIT (Unified Auditing), *1-1*
CALL, *1-1*
COMMENT, *1-1*
COMMIT, *1-1*
CREATE ANALYTIC VIEW, *1-1*
CREATE ATTRIBUTE DIMENSION, *1-1*
CREATE AUDIT POLICY, *1-1*
CREATE CLUSTER, *1-1*
CREATE CONTEXT, *1-1*
CREATE CONTROLFILE, *1-1*
CREATE DATABASE, *1-1*
CREATE DATABASE LINK, *1-1*
CREATE DIMENSION, *1-1*
CREATE DIRECTORY, *1-1*
CREATE DISKGROUP, *1-1*
CREATE EDITION, *1-1*
CREATE FLASHBACK ARCHIVE, *1-1*
CREATE FUNCTION, *1-1*
CREATE HIERARCHY, *1-1*
CREATE INDEX, *1-1*
CREATE INDEXTYPE, *1-1*
CREATE INMEMORY JOIN GROUP, *1-1*
CREATE JAVA, *1-1*
CREATE LIBRARY, *1-1*
CREATE LOCKDOWN PROFILE, *1-1*
CREATE MATERIALIZED VIEW, *1-1*
CREATE MATERIALIZED VIEW LOG, *1-1*
CREATE MATERIALIZED ZONEMAP, *1-1*
CREATE OPERATOR, *1-1*
CREATE OUTLINE, *1-1*
CREATE PACKAGE, *1-1*
CREATE PACKAGE BODY, *1-1*
CREATE PFILE, *1-1*
CREATE PLUGGABLE DATABASE, *1-1*
CREATE PROCEDURE, *1-1*
CREATE PROFILE, *1-1*
CREATE PROPERTY GRAPH, *1-1*
CREATE RESTORE POINT, *1-1*
CREATE ROLE, *1-1*
CREATE ROLLBACK SEGMENT, *1-1*
CREATE SCHEMA, *1-1*
CREATE SEQUENCE, *1-1*
CREATE SPFILE, *1-1*

SQL statements *(continued)*
CREATE SYNONYM, *1-1*
CREATE TABLE, *1-1*
CREATE TABLESPACE, *1-1*
CREATE TABLESPACE SET, *1-1*
CREATE TRIGGER, *1-1*
CREATE TYPE, *1-1*
CREATE TYPE BODY, *1-1*
CREATE USER, *1-1*
CREATE VECTOR INDEX, *1-1*
CREATE VIEW, *1-1*
DELETE, *1-1*
DISASSOCIATE STATISTICS, *1-1*
DROP ANALYTIC VIEW, *1-1*
DROP ATTRIBUTE DIMENSION, *1-1*
DROP AUDIT POLICY, *1-1*
DROP CLUSTER, *1-1*
DROP CONTEXT, *1-1*
DROP DATABASE, *1-1*
DROP DATABASE LINK, *1-1*
DROP DIMENSION, *1-1*
DROP DIRECTORY, *1-1*
DROP DISKGROUP, *1-1*
DROP EDITION, *1-1*
DROP FLASHBACK ARCHIVE, *1-1*
DROP FUNCTION, *1-1*
DROP HIERARCHY, *1-1*
DROP INDEX, *1-1*
DROP INDEXTYPE, *1-1*
DROP INMEMORY JOIN GROUP, *1-1*
DROP JAVA, *1-1*
DROP LIBRARY, *1-1*
DROP LOCKDOWN PROFILE, *1-1*
DROP MATERIALIZED VIEW, *1-1*
DROP MATERIALIZED VIEW LOG, *1-1*
DROP MATERIALIZED ZONEMAP, *1-1*
DROP OPERATOR, *1-1*
DROP OUTLINE, *1-1*
DROP PACKAGE, *1-1*
DROP PLUGGABLE DATABASE, *1-1*
DROP PROCEDURE, *1-1*
DROP PROFILE, *1-1*
DROP PROPERTY GRAPH, *1-1*
DROP RESTORE POINT, *1-1*
DROP ROLE, *1-1*
DROP ROLLBACK SEGMENT, *1-1*
DROP SEQUENCE, *1-1*
DROP SYNONYM, *1-1*
DROP TABLE, *1-1*
DROP TABLESPACE, *1-1*
DROP TABLESPACE SET, *1-1*
DROP TRIGGER, *1-1*
DROP TYPE, *1-1*
DROP TYPE BODY, *1-1*
DROP USER, *1-1*
DROP VIEW, *1-1*

SQL statements *(continued)*
EXPLAIN PLAN, *1-1*
FLASHBACK DATABASE, *1-1*
FLASHBACK TABLE, *1-1*
GRANT, *1-1*
INSERT, *1-1*
LOCK TABLE, *1-1*
MERGE, *1-1*
NOAUDIT (Traditional Auditing), *1-1*
NOAUDIT (Unified Auditing), *1-1*
PURGE, *1-1*
RENAME, *1-1*
REVOKE, *1-1*
ROLLBACK, *1-1*
SAVEPOINT, *1-1*
SELECT, *1-1*
SET CONSTRAINT, *1-1*
SET ROLE, *1-1*
SET TRANSACTION, *1-1*
TRUNCATE CLUSTER, *1-1*
TRUNCATE TABLE, *1-1*
UPDATE, *1-1*
sql_format of TO_DSINTERVAL function, *5-1*
SQL*Plus commands, *A-1*
@ (at sign), *A-3*
/ (slash), *A-4*
APPEND, *A-3*
CHANGE, *A-3*
CONNECT, *A-3*
DEL, *A-3*
DESCRIBE, *A-3*
DISCONNECT, *A-4*
EDIT, *A-3*
EXECUTE, *A-4*
EXIT, *A-4*
GET, *A-3*
HELP, *A-1*
HOST, *A-1*
INPUT, *A-3*
LIST, *A-3*
QUIT, *A-4*
RUN, *A-4*
SAVE, *A-3*
SET, *A-2*
SHOW, *A-2*
SHUTDOWN, *A-4*
SPOOL, *A-3*
SQLPLUS, *A-1*
START, *A-3*
STARTUP, *A-2*
SQL/DS data types
restrictions on, *6-6*
SQLPLUS SQL*Plus command, *A-1*
SQRT function, *2-1*
standard_actions, *5-1*
STANDARD_HASH function, *2-1*
standby_database_clauses, *5-1*
standbys_clause, *5-1*
START SQL*Plus command, *A-3*
start_standby_clause, *5-1*
STARTUP SQL*Plus command, *A-2*
startup_clauses, *5-1*
statement_clauses, *5-1*
statements, *1-1*
see also SQL statements, *1-1*
STATS_BINOMIAL_TEST function, *2-1*
STATS_CROSSTAB function, *2-1*
STATS_F_TEST function, *2-1*
STATS_KS_TEST function, *2-1*
STATS_MODE function, *2-1*
STATS_MW_TEST function, *2-1*
STATS_ONE_WAY_ANOVA function, *2-1*
STATS_T_TEST_INDEP function, *2-1*
STATS_T_TEST_INDEPU function, *2-1*
STATS_T_TEST_ONE function, *2-1*
STATS_T_TEST_PAIRED function, *2-1*
STATS_WSR_TEST function, *2-1*
STDDEV function, *2-1*
STDDEV_POP function, *2-1*
STDDEV_SAMP function, *2-1*
still_image_object_types, *5-1*
stop_standby_clause, *5-1*
storage_clause, *5-1*
storage_table_clause, *5-1*
string, *5-1*
striping_clause, *5-1*
SUBMULTISET condition, *4-1*
subpartition_by_hash, *5-1*
subpartition_by_list, *5-1*
subpartition_by_range, *5-1*
subpartition_extended_name, *5-1*
subpartition_extended_names, *5-1*
subpartition_or_key_value, *5-1*
subpartition_spec, *5-1*
subpartition_template, *5-1*
subquery, *5-1*
subquery_factoring_clause, *5-1*
subquery_restriction_clause, *5-1*
substitutable_column_clause, *5-1*
SUBSTR function, *2-1*
SUM function, *2-1*
supplemental_db_logging, *5-1*
supplemental_id_key_clause, *5-1*
supplemental_log_grp_clause, *5-1*
supplemental_logging_props, *5-1*
supplemental_plsql_clause, *5-1*
supplemental_table_logging, *5-1*
supplied data types, *6-1*, *6-5*
switch_logfile_clause, *5-1*
switchover_clause, *5-1*
syntax for subclauses, *5-1*
SYS_CONNECT_BY_PATH function, *2-1*

SYS_CONTEXT function, *2-1*
SYS_DBURIGEN function, *2-1*
SYS_EXTRACT_UTC function, *2-1*
SYS_GUID function, *2-1*
SYS_OP_ZONE_ID function, *2-1*
SYS_TYPEID function, *2-1*
SYS_XMLAGG function, *2-1*
SYS_XMLGEN function, *2-1*
SYSDATE function, *2-1*
system_partitioning, *5-1*
SYSTIMESTAMP function, *2-1*

T

table_collection_expression, *5-1*
table_compression, *5-1*
table_index_clause, *5-1*
table_partition_description, *5-1*
table_partitioning_clauses, *5-1*
table_properties, *5-1*
table_reference, *5-1*
tablespace_clauses, *5-1*
tablespace_datafile_clauses, *5-1*
tablespace_encryption_clause, *5-1*
tablespace_encryption_spec, *5-1*
tablespace_group_clause, *5-1*
tablespace_logging_clauses, *5-1*
tablespace_retention_clause, *5-1*
tablespace_state_clauses, *5-1*
TAN function, *2-1*
TANH function, *2-1*
tempfile_reuse_clause, *5-1*
temporary_tablespace_clause, *5-1*
TIME data type
  DB2, *6-6*
  SQL/DS, *6-6*
time format models, *7-6*
time zone formatting, *7-6*
TIME_BUCKET(datetime)ti, *2-1*
timeout_clause, *5-1*
TIMESTAMP data type
  DB2, *6-6*
  SQL/DS, *6-6*
TIMESTAMP_TO_SCN function, *2-1*
TO_APPROX_COUNT_DISTINCT function, *2-1*
TO_APPROX_PERCENTILE function, *2-1*
TO_BINARY_DOUBLE function, *2-1*
TO_BINARY_FLOAT function, *2-1*
TO_BLOB (bfile) function, *2-1*
TO_BLOB (raw) function, *2-1*
TO_CHAR (bfile|blob) function, *2-1*
TO_CHAR (character) function, *2-1*
TO_CHAR (datetime) function, *2-1*
TO_CHAR (number) function, *2-1*
TO_CLOB (bfile|blob) function, *2-1*
TO_CLOB (character) function, *2-1*
TO_DATE function, *2-1*
TO_DSINTERVAL function, *2-1*
TO_LOB function, *2-1*
TO_MULTI_BYTE function, *2-1*
TO_NCHAR (character) function, *2-1*
TO_NCHAR (datetime) function, *2-1*
TO_NCHAR (number) function, *2-1*
TO_NCLOB function, *2-1*
TO_NUMBER function, *2-1*
TO_SINGLE_BYTE function, *2-1*
TO_TIMESTAMP function, *2-1*
TO_TIMESTAMP_TZ function, *2-1*
TO_YMINTERVAL function, *2-1*
trace_file_clause, *5-1*
TRANSLATE function, *2-1*
TRANSLATE...USING function, *2-1*
TREAT function, *2-1*
TRIM function, *2-1*
TRUNC (date) function, *2-1*
TRUNC (number) function, *2-1*
TRUNCATE CLUSTER statement, *1-1*
TRUNCATE TABLE statement, *1-1*
truncate_partition_subpart, *5-1*
ts_file_name_convert, *5-1*
type constructor expressions, *3-1*
TZ_OFFSET function, *2-1*

U

UID function, *2-1*
UNDER_PATH condition, *4-1*
undo_mode_clause, *5-1*
undo_tablespace, *5-1*
undo_tablespace_clause, *5-1*
undrop_disk_clause, *5-1*
UNISTR function, *2-1*
unpivot_clause, *5-1*
unpivot_in_clause, *5-1*
unusable_editions_clause, *5-1*
UPDATE statement, *1-1*
update_all_indexes_clause, *5-1*
update_global_index_clause, *5-1*
update_index_clauses, *5-1*
update_index_partition, *5-1*
update_index_subpartition, *5-1*
update_set_clause, *5-1*
upgrade_table_clause, *5-1*
UPPER function, *2-1*
use_key, *5-1*
USER function, *2-1*
user_clauses, *5-1*
user_tablespaces_clause, *5-1*
user-defined data types, *6-1*
user-defined functions, *2-1*
USERENV function, *2-1*
usergroup_clauses, *5-1*

using_clause, *5-1*
using_function_clause, *5-1*
using_index_clause, *5-1*
using_statistics_type, *5-1*
using_type_clause, *5-1*

V

VALIDATE_CONVERSION function, *2-1*
validation_clauses, *5-1*
VALUE function, *2-1*
values_clause, *5-1*
VAR_POP function, *2-1*
VAR_SAMP function, *2-1*
VARGRAPHIC data type
    DB2, *6-6*
    SQL/DS, *6-6*
VARIANCE function, *2-1*
varray_col_properties, *5-1*
varray_storage_clause, *5-1*
virtual_column_definition, *5-1*
VSIZE function, *2-1*

W

where_clause, *5-1*
WIDTH_BUCKET function, *2-1*
window_clause, *5-1*
window_expression, *5-1*
windowing_clause, *5-1*
with_clause, *5-1*

X

XML_attributes_clause, *5-1*
XML_passing_clause, *5-1*
XML_table_column, *5-1*
XML_types, *6-5*
XMLAGG function, *2-1*
XMLCast function, *2-1*
XMLCDATA function, *2-1*
XMLCOLATTVAL function, *2-1*
XMLCOMMENT function, *2-1*
XMLCONCAT function, *2-1*
XMLDIFF function, *2-1*
XMLELEMENT function, *2-1*
XMLEXISTS function, *2-1*
XMLFOREST function, *2-1*
XMLIndex_clause, *5-1*
XMLISVALID function, *2-1*
XMLnamespaces_clause, *5-1*
XMLPARSE function, *2-1*
XMLPATCH function, *2-1*
XMLPI function, *2-1*
XMLQUERY function, *2-1*
XMLROOT function, *2-1*
XMLSchema_spec, *5-1*
XMLSEQUENCE function, *2-1*
XMLSERIALIZE function, *2-1*
XMLTABLE function, *2-1*
XMLTABLE_options, *5-1*
XMLTRANSFORM function, *2-1*
XMLType_column_properties, *5-1*
XMLType_storage, *5-1*
XMLType_table, *5-1*
XMLType_view_clause, *5-1*
XMLType_virtual_columns, *5-1*

Y

ym_iso_format of TO_YMINTERVAL function, *5-1*

Z

zonemap_attributes, *5-1*
zonemap_clause, *5-1*
zonemap_refresh_clause, *5-1*