Oracle® Database

SQL Quick Reference

10*g* Release 2 (10.2)

B14195-01

June 2005

Oracle Database SQL Quick Reference, 10*g* Release 2 (10.2)

B14195-01

Contributors: Joan Gregoire, Diana Lorentz, Cathy Shea, Simon Watt

Contents

Preface

This quick reference contains a high-level 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 Standards Organization (ISO) standard.

This Preface contains these topics:

- Audience
- Documentation Accessibility
- Related Documents
- Conventions

Audience

SQL Quick Reference is intended for all users of Oracle SQL.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at

`http://www.oracle.com/accessibility/`

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

TTY Access to Oracle Support Services

Oracle provides dedicated Text Telephone (TTY) access to Oracle Support Services within the United States of America 24 hours a day, seven days a week. For TTY support, call 800.446.2398.

Related Documents

For more information, see these Oracle resources:

- *Oracle Database SQL Reference*
- *PL/SQL User's Guide and Reference*
- *SQL*Plus User's Guide and Reference*

Conventions

The following text conventions are used in this document:

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

1 SQL Statements

This chapter presents the syntax for Oracle SQL statements.

This chapter includes the following section:

- Syntax for SQL Statements

Syntax for SQL Statements

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

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

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

ALTER CLUSTER

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

ALTER DATABASE

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

ALTER DIMENSION

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

ALTER DISKGROUP

```
ALTER DISKGROUP
    { diskgroup_name
        { add_disk_clause | drop_disk_clause }
            [, { add_disk_clause | drop_disk_clause } ]...
        | resize_disk_clauses
        } [ rebalance_diskgroup_clause ]
        | {rebalance_diskgroup_clause
          | check_diskgroup_clause
          | diskgroup_template_clauses
          | diskgroup_directory_clauses
          | diskgroup_alias_clauses
          | drop_diskgroup_file_clause
    }
    | { diskgroup_name
          [, diskgroup_name ]...
      | ALL
      }
      { undrop_disk_clause
      | diskgroup_availability
      }
    }
```

ALTER FUNCTION

```
ALTER FUNCTION [ schema. ]function
  COMPILE [ DEBUG ]
  [ compiler_parameters_clause
    [ compiler_parameters_clause ] ... ]
  [ REUSE SETTINGS ] ;
```

ALTER INDEX

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

ALTER INDEXTYPE

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

ALTER JAVA

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

ALTER MATERIALIZED VIEW

```
ALTER MATERIALIZED VIEW
  [ schema. ](materialized_view)
  [ physical_attributes_clause
  | table_compression
  | LOB_storage_clause
    [, LOB_storage_clause ]...
  | modify_LOB_storage_clause
    [, modify_LOB_storage_clause ]...
  | alter_table_partitioning
  | parallel_clause
  | logging_clause
  | allocate_extent_clause
  | shrink_clause
  | { CACHE | NOCACHE }
  ]
  [ alter_iot_clauses ]
  [ USING INDEX physical_attributes_clause ]
  [ MODIFY scoped_table_ref_constraint
  | alter_mv_refresh
  ]
  [ { ENABLE | DISABLE } QUERY REWRITE
  | COMPILE
  | CONSIDER FRESH
  ] ;
```

ALTER MATERIALIZED VIEW LOG

```
ALTER MATERIALIZED VIEW LOG [ FORCE ]
  ON [ schema. ]table
  [ physical_attributes_clause
  | alter_table_partitioning
  | parallel_clause
  | logging_clause
  | allocate_extent_clause
  | shrink_clause
  | { CACHE | NOCACHE }
  ]
  [ ADD
      { { OBJECT ID
        | PRIMARY KEY
        | ROWID
        | SEQUENCE
        }
        [ (column [, column ]...) ]
      | (column [, column ]... )
      }
        [, { { OBJECT ID
             | PRIMARY KEY
             | ROWID
             | SEQUENCE
             }
             [ (column [, column ]...) ]
           | (column [, column ]...)
           }
        ]...
      [ new_values_clause ]
  ] ;
```

ALTER OPERATOR

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

ALTER OUTLINE

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

ALTER PACKAGE

```
ALTER PACKAGE [ schema. ]package
  COMPILE [ DEBUG ]
  [ PACKAGE | SPECIFICATION | BODY ]
  [ compiler_parameters_clause
    [ compiler_parameters_clause ] ... ]
  [ REUSE SETTINGS ] ;
```

ALTER PROCEDURE

```
ALTER PROCEDURE [ schema. ]procedure
  COMPILE [ DEBUG ]
  [ compiler_parameters_clause
    [ compiler_parameters_clause ] ... ]
  [ REUSE SETTINGS ] ;
```

ALTER PROFILE

```
ALTER PROFILE profile LIMIT
  { resource_parameters | password_parameters }
    [ resource_parameters | password_parameters
    ]... ;
```

ALTER RESOURCE COST

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

ALTER ROLLBACK SEGMENT

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

ALTER SEQUENCE

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

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
    }
  | alter_session_set_clause
  } ;
```

ALTER SYSTEM

```
ALTER SYSTEM
  { archive_log_clause
  | checkpoint_clause
  | check_datafiles_clause
  | distributed_recov_clauses
  | FLUSH { SHARED_POOL | BUFFER_CACHE }
  | end_session_clauses
  | SWITCH LOGFILE
  | { SUSPEND | RESUME }
  | quiesce_clauses
  | alter_system_security_clauses
  | shutdown_dispatcher_clause
  | REGISTER
  | SET alter_system_set_clause
        [ alter_system_set_clause ]...
  | RESET alter_system_reset_clause
          [ alter_system_reset_clause ]...
  } ;
```

ALTER TABLE

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

ALTER TABLESPACE

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

ALTER TRIGGER

```
ALTER TRIGGER [ schema. ]trigger
  { ENABLE
  | DISABLE
  | RENAME TO new_name
  | COMPILE [ DEBUG ]
      [ compiler_parameters_clause
        [ compiler_parameters_clause ] ... ]
      [ REUSE SETTINGS ]
  } ;
```

ALTER TYPE

```
ALTER TYPE [ schema. ]type
  { compile_type_clause
  | replace_type_clause
  | { alter_method_spec
    | alter_attribute_definition
    | alter_collection_clauses
    | [ NOT ] { INSTANTIABLE | FINAL }
    }
    [ dependent_handling_clause ]
  } ;
```

ALTER USER

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

ALTER VIEW

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

ANALYZE

```
ANALYZE
  { TABLE [ schema. ]table
      [ PARTITION (partition)
      | SUBPARTITION (subpartition)
      ]
  | INDEX [ schema. ]index
      [ PARTITION (partition)
      | SUBPARTITION (subpartition)
      ]
  | CLUSTER [ schema. ]cluster
  }
  { validation_clauses
  | LIST CHAINED ROWS [ into_clause ]
  | DELETE [ SYSTEM ] STATISTICS
  | compute_statistics_clause
  | estimate_statistics_clause
  } ;
```

ASSOCIATE STATISTICS

```
ASSOCIATE STATISTICS WITH
  { column_association | function_association } ;
```

AUDIT

```
AUDIT
  { sql_statement_clause | schema_object_clause | NETWORK }
  [ BY { SESSION | ACCESS } ]
  [ WHENEVER [ NOT ] SUCCESSFUL ] ;
```

CALL

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

COMMENT

```
COMMENT ON
  { TABLE [ schema. ]
    { table | view }
  | COLUMN [ schema. ]
    { table. | view. | materialized_view. } column
  | OPERATOR [ schema. ] operator
  | INDEXTYPE [ schema. ] indextype
  | MATERIALIZED VIEW materialized_view
  }
  IS string ;
```

COMMIT

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

CREATE CLUSTER

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

```
          HASHKEYS integer [ HASH IS expr ]
        }
      ]...
  ]
  [ parallel_clause ]
  [ NOROWDEPENDENCIES | ROWDEPENDENCIES ]
  [ CACHE | NOCACHE ] ;
```

CREATE CONTEXT

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

CREATE CONTROLFILE

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

CREATE DATABASE

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

CREATE DATABASE LINK

```
CREATE [ SHARED ] [ PUBLIC ] DATABASE LINK dblink
  [ CONNECT TO
    { CURRENT_USER
```

```
      | user IDENTIFIED BY password [ dblink_authentication ]
      }
    | dblink_authentication
    ]
      [ CONNECT TO
        { CURRENT_USER
        | user IDENTIFIED BY password [ dblink_authentication ]
        }
      | dblink_authentication
      ]...
    [ USING connect_string ] ;
```

CREATE DIMENSION

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

CREATE DIRECTORY

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

CREATE DISKGROUP

```
CREATE DISKGROUP diskgroup_name
  [ { HIGH | NORMAL | EXTERNAL } REDUNDANCY ]
  [ FAILGROUP failgroup_name ]
  DISK qualified_disk_clause
       [, qualified_disk_clause ]...
    [ [ FAILGROUP failgroup_name ]
      DISK qualified_disk_clause
           [, qualified_disk_clause ]...
    ]... ;
```

CREATE FUNCTION

```
CREATE [ OR REPLACE ] FUNCTION [ schema. ]function
  [ (argument [ IN | OUT | IN OUT ]
     [ NOCOPY ] datatype
       [, argument [ IN | OUT | IN OUT ]
          [ NOCOPY ] datatype
       ]...
    )
  ]
  RETURN datatype
  [ { invoker_rights_clause
    | DETERMINISTIC
    | parallel_enable_clause
    }
      [ invoker_rights_clause
      | DETERMINISTIC
      | parallel_enable_clause
      ]...
  ]
  { { AGGREGATE | PIPELINED }
    USING [ schema. ]implementation_type
  | [ PIPELINED ]
    { IS | AS }
    { pl/sql_function_body | call_spec }
  } ;
```

CREATE INDEX

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

CREATE INDEXTYPE

```
CREATE [ OR REPLACE ] INDEXTYPE
  [ schema. ]indextype FOR
  [ schema. ]operator (paramater_type
                       [, paramater_type ]...)
    [, [ schema. ]operator (paramater_type
                            [, paramater_type ]...)
    ]...
  using_type_clause ;
```

CREATE JAVA

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

CREATE LIBRARY

```
CREATE [ OR REPLACE ] LIBRARY [ schema. ]libname
  { IS | AS } 'filename' [ AGENT 'agent_dblink' ] ;
```

CREATE MATERIALIZED VIEW

```
CREATE MATERIALIZED VIEW
  [ schema. ]materialized_view
  [ OF [ schema. ]object_type ]
  [ (scoped_table_ref_constraint) ]
  { ON PREBUILT TABLE
    [ { WITH | WITHOUT } REDUCED PRECISION ]
  | physical_properties materialized_view_props
  }
  [ USING INDEX
    [ physical_attributes_clause
    | TABLESPACE tablespace
    ]
      [ physical_attributes_clause
      | TABLESPACE tablespace
      ]...
  | USING NO INDEX
  ]
  [ create_mv_refresh ]
  [ FOR UPDATE ]
  [ { DISABLE | ENABLE }
    QUERY REWRITE
  ]
  AS subquery ;
```

CREATE MATERIALIZED VIEW LOG

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

CREATE OPERATOR

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

CREATE OUTLINE

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

CREATE PACKAGE

```
CREATE [ OR REPLACE ] PACKAGE [ schema. ]package
   [ invoker_rights_clause ]
   { IS | AS } pl/sql_package_spec ;
```

CREATE PACKAGE BODY

```
CREATE [ OR REPLACE ] PACKAGE BODY
```

```
   [ schema. ]package
   { IS | AS } pl/sql_package_body ;
```

CREATE PFILE

```
CREATE PFILE [= 'pfile_name' ]
   FROM SPFILE [= 'spfile_name'] ;
```

CREATE PROCEDURE

```
CREATE [ OR REPLACE ] PROCEDURE [ schema. ]procedure
   [ (argument [ { IN | OUT | IN OUT } ]
               [ NOCOPY ]
               datatype [ DEFAULT expr ]
       [, argument [ { IN | OUT | IN OUT } ]
                   [ NOCOPY ]
                   datatype [ DEFAULT expr ]
       ]...
     )
   ]
   [ invoker_rights_clause ]
   { IS | AS }
   { pl/sql_subprogram_body | call_spec } ;
```

CREATE PROFILE

```
CREATE PROFILE profile
   LIMIT { resource_parameters
         | password_parameters
         }
           [ resource_parameters
           | password_parameters
           ]... ;
```

CREATE RESTORE POINT

```
CREATE RESTORE POINT restore_point
   [ GUARANTEE FLASHBACK DATABASE ];
```

CREATE ROLE

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

CREATE ROLLBACK SEGMENT

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

CREATE SCHEMA

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

CREATE SEQUENCE

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

CREATE SPFILE

```
CREATE SPFILE [= 'spfile_name' ]
  FROM PFILE [= 'pfile_name' ] ;
```

CREATE SYNONYM

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

CREATE TABLE

```
{ relational_table | object_table | XMLType_table }
```

CREATE TABLESPACE

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

CREATE TRIGGER

```
CREATE [ OR REPLACE ] TRIGGER [ schema. ]trigger
   { BEFORE | AFTER | INSTEAD OF }
   { dml_event_clause
   | { ddl_event [ OR ddl_event ]...
     | database_event [ OR database_event ]...
     }
     ON { [ schema. ]SCHEMA
        | DATABASE
        }
   }
   [ WHEN (condition) ]
   { pl/sql_block | call_procedure_statement } ;
```

CREATE TYPE

```
{ create_incomplete_type
| create_object_type
| create_varray_type
| create_nested_table_type
}
```

CREATE TYPE BODY

```
CREATE [ OR REPLACE ] TYPE BODY [ schema. ]type_name
   { IS | AS }
   { subprogram_declaration
   | map_order_func_declaration
   }
     [, { subprogram_declaration
        | map_order_func_declaration
        }
     ]...
   END ;
```

CREATE USER

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

CREATE VIEW

```
CREATE [ OR REPLACE ] [ [ NO ] FORCE ] VIEW
   [ schema. ]view
   [ (alias [ inline_constraint
              [ inline_constraint ]... ]
     | out_of_line_constraint
       [, alias [ inline_constraint
                  [ inline_constraint ]... ]
       | out_of_line_constraint
       ]...
     )
   | object_view_clause
   | XMLType_view_clause
   ]
   AS subquery [ subquery_restriction_clause ] ;
```

DELETE

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

DISASSOCIATE STATISTICS

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

DROP CLUSTER

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

DROP CONTEXT

```
DROP CONTEXT namespace ;
```

DROP DATABASE

```
DROP DATABASE ;
```

DROP DATABASE LINK

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

DROP DIMENSION

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

DROP DIRECTORY

```
DROP DIRECTORY directory_name ;
```

DROP DISKGROUP

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

DROP FUNCTION

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

DROP INDEX

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

DROP INDEXTYPE

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

DROP JAVA

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

DROP LIBRARY

```
DROP LIBRARY library_name ;
```

DROP MATERIALIZED VIEW

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

DROP MATERIALIZED VIEW LOG

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

DROP OPERATOR

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

DROP OUTLINE

```
DROP OUTLINE outline ;
```

DROP PACKAGE

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

DROP PROCEDURE

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

DROP PROFILE

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

DROP RESTORE POINT

```
DROP RESTORE POINT restore_point ;
```

DROP ROLE

```
DROP ROLE role ;
```

DROP ROLLBACK SEGMENT

```
DROP ROLLBACK SEGMENT rollback_segment ;
```

DROP SEQUENCE

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

DROP SYNONYM

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

DROP TABLE

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

DROP TABLESPACE

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

DROP TRIGGER

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

DROP TYPE

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

DROP TYPE BODY

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

DROP USER

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

DROP VIEW

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

EXPLAIN PLAN

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

FLASHBACK DATABASE

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

FLASHBACK TABLE

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

GRANT

```
GRANT { grant_system_privileges
      | grant_object_privileges
      } ;
```

INSERT

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

LOCK TABLE

```
LOCK TABLE
   [ schema. ] { table | view }
   [ { PARTITION (partition)
     | SUBPARTITION (subpartition)
     }
   | @ dblink
   ]
     [, [ schema. ] { table | view }
        [ { PARTITION (partition)
          | SUBPARTITION (subpartition)
          }
        | @ dblink
        ]
     ]...
   IN lockmode MODE
   [ NOWAIT ] ;
```

MERGE

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

NOAUDIT

```
NOAUDIT
   { sql_statement_clause
   | schema_object_clause
   | NETWORK
   }
   [ WHENEVER [ NOT ] SUCCESSFUL ] ;
```

PURGE

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

RENAME

```
RENAME old_name
   TO new_name ;
```

REVOKE

```
REVOKE { revoke_system_privileges
       | revoke_object_privileges
       } ;
```

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

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

UPDATE

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

2 SQL Functions

This chapter presents the syntax for SQL functions.

This chapter includes the following section:

- Syntax for SQL Functions

Syntax for SQL Functions

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

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

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

ABS

```
ABS(n)
```

ACOS

```
ACOS(n)
```

ADD_MONTHS

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

analytic_function

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

APPENDCHILDXML

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

ASCII

```
ASCII(char)
```

ASCIISTR

```
ASCIISTR(char)
```

ASIN

```
ASIN(n)
```

ATAN

```
ATAN(n)
```

ATAN2

```
ATAN2(n1 { , | / } n2)
```

AVG

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

BFILENAME

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

BIN_TO_NUM

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

BITAND

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

CARDINALITY

```
CARDINALITY(nested_table)
```

CAST

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

CEIL

```
CEIL(n)
```

CHARTOROWID

```
CHARTOROWID(char)
```

CHR

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

CLUSTER_ID

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

CLUSTER_PROBABILITY

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

CLUSTER_SET

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

COALESCE

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

COLLECT

```
COLLECT (column)
```

COMPOSE

```
COMPOSE(char)
```

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
       | TWO_SIDED_SIG
       }
    ]
   )
```

COS

```
COS(n)
```

COSH

```
COSH(n)
```

COUNT

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

COVAR_POP

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

COVAR_SAMP

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

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 ])
```

DBTIMEZONE

```
DBTIMEZONE
```

DECODE

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

DECOMPOSE

```
DECOMPOSE( string [ CANONICAL | COMPATIBILITY ] )
```

DELETXML

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

DENSE_RANK (aggregate)

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

DENSE_RANK (aggregate)

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

DEPTH

```
DEPTH(correlation_integer)
```

DEREF

```
DEREF(expr)
```

DUMP

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

EMPTY_BLOB, EMPTY_CLOB

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

EXISTSNODE

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

EXP

```
EXP(n)
```

EXTRACT (datetime)

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

EXTRACT (XML)

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

EXTRACTVALUE

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

FEATURE_ID

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

FEATURE_SET

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

FEATURE_VALUE

```
FEATURE_VALUE ( [ schema . ] model
   [ , feature_id ] mining_attribute_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 [ IGNORE NULLS ])
   OVER (analytic_clause)
```

FLOOR

```
FLOOR(n)
```

FROM_TZ

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

GREATEST

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

GROUP_ID

```
GROUP_ID( )
```

GROUPING

```
GROUPING(expr)
```

GROUPING_ID

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

HEXTORAW

```
HEXTORAW(char)
```

INITCAP

```
INITCAP(char)
```

INSERTCHILDXML

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

INSERTXMLBEFORE

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

INSTR

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

ITERATION_NUMBER

```
ITERATION_NUMBER
```

LAG

```
LAG(value_expr [, 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 [ IGNORE NULLS ])
   OVER (analytic_clause)
```

LEAD

```
LEAD(value_expr [, offset ] [, default ])
   OVER ([ query_partition_clause ] order_by_clause)
```

LEAST

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

LENGTH

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

LN

```
LN(n)
```

LNNVL

```
LNNVL(condition)
```

LOCALTIMESTAMP

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

LOG

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

LOWER

```
LOWER(char)
```

LPAD

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

LTRIM

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

MAKE_REF

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

MAX

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

MEDIAN

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

MIN

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

MOD

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

MONTHS_BETWEEN

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

NANVL

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

NCHR

```
NCHR(number)
```

NEW_TIME

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

NEXT_DAY

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

NLS_CHARSET_DECL_LEN

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

NLS_CHARSET_ID

```
NLS_CHARSET_ID ( string )
```

NLS_CHARSET_NAME

```
NLS_CHARSET_NAME(number)
```

NLS_INITCAP

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

NLS_LOWER

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

NLS_UPPER

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

NLSSORT

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

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_HASH

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

PATH

```
PATH (correlation_integer)
```

PERCENT_RANK (aggregate)

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

PERCENT_RANK (analytic)

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

PERCENTILE_CONT

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

PERCENTILE_DISC

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

POWER

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

POWERMULTISET

```
POWERMULTISET(expr)
```

POWERMULTISET_BY_CARDINALITY

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

PREDICTION

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

PREDICTION_COST

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

PREDICTION_DETAILS

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

PREDICTION_PROBABILITY

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

PREDICTION_SET

```
PREDICTION_SET ( [ schema . ] model [ , bestN [ , cutoff ] ]
  [ cost_matrix_clause ] mining_attribute_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_INSTR

```
REGEXP_INSTR (source_char, pattern
              [, position
                 [, occurrence
                    [, return_option
                       [, match_parameter ]
                    ]
                 ]
              ]
             )
```

REGEXP_REPLACE

```
REGEXP_REPLACE(source_char, pattern
               [, replace_string
                  [, position
                     [, occurrence
                        [, match_parameter ]
                     ]
                  ]
               ]
              )
```

REGEXP_SUBSTR

```
REGEXP_SUBSTR(source_char, pattern
              [, position
                 [, occurrence
                    [, match_parameter ]
                 ]
              ]
             )
```

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

```
{ REGR_SLOPE
| REGR_INTERCEPT
| REGR_COUNT
| REGR_R2
| REGR_AVGX
| REGR_AVGY
| REGR_SXX
| REGR_SYY
| REGR_SXY
}
```

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

REMAINDER

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

REPLACE

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

ROUND (date)

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

ROUND (number)

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

ROW_NUMBER

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

ROWIDTOCHAR

```
ROWIDTOCHAR(rowid)
```

ROWIDTONCHAR

```
ROWIDTONCHAR(rowid)
```

RPAD

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

RTRIM

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

SCN_TO_TIMESTAMP

```
SCN_TO_TIMESTAMP(number)
```

SESSIONTIMEZONE

```
SESSIONTIMEZONE
```

SET

```
SET (nested_table)
```

SIGN

```
SIGN(n)
```

SIN

```
SIN(n)
```

SINH

```
SINH(n)
```

SOUNDEX

```
SOUNDEX(char)
```

SQRT

```
SQRT(n)
```

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

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_TYPEID

```
SYS_TYPEID(object_type_value)
```

SYS_XMLAGG

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

SYS_XMLGEN

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

SYSDATE

```
SYSDATE
```

SYSTIMESTAMP

```
SYSTIMESTAMP
```

TAN

```
TAN(n)
```

TANH

```
TANH(n)
```

TIMESTAMP_TO_SCN

```
TIMESTAMP_TO_SCN(timestamp)
```

TO_BINARY_DOUBLE

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

TO_BINARY_FLOAT

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

TO_CHAR (character)

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

TO_CHAR (datetime)

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

TO_CHAR (number)

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

TO_CLOB

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

TO_DATE

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

TO_DSINTERVAL

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

TO_LOB

```
TO_LOB(long_column)
```

TO_MULTI_BYTE

```
TO_MULTI_BYTE(char)
```

TO_NCHAR (character)

```
TO_NCHAR({char | clob | nclob}
         [, fmt [, 'nlsparam' ] ]
        )
```

TO_NCHAR (datetime)

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

TO_NCHAR (number)

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

TO_NCLOB

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

TO_NUMBER

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

TO_SINGLE_BYTE

```
TO_SINGLE_BYTE(char)
```

TO_TIMESTAMP

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

TO_TIMESTAMP_TZ

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

TO_YMINTERVAL

```
TO_YMINTERVAL(char)
```

TRANSLATE

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

TRANSLATE ... USING

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

TREAT

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

TRIM

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

TRUNC (date)

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

TRUNC (number)

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

TZ_OFFSET

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

UID

```
UID
```

UNISTR

```
UNISTR( string )
```

UPDATEXML

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

UPPER

```
UPPER(char)
```

USER

```
USER
```

user-defined function

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

USERENV

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

VALUE

```
VALUE(correlation_variable)
```

VAR_POP

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

VAR_SAMP

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

VARIANCE

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

VSIZE

```
VSIZE(expr)
```

WIDTH_BUCKET

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

XMLAGG

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

XMLCOLATTVAL

```
XMLCOLATTVAL
  (value_expr [ AS c_alias ]
    [, value_expr [ AS c_alias ]
      ]...
  )
```

XMLCOMMENT

```
XMLCOMMENT ( value_expr )
```

XMLCDATA

```
 XMLCDATA ( value_expr )
```

XMLCONCAT

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

XMLELEMENT

```
XMLELEMENT
 ( [ NAME ] identifier
    [, XML_attributes_clause ]
    [, value_expr [ AS c_alias ]
      [, value_expr [ AS c_alias ]
    ]...
 )
```

XMLFOREST

```
XMLFOREST
  ( value_expr [ AS c_alias ]
    [, value_expr [ AS c_alias ]
      ]...
  )
```

XMLPARSE

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

XMLPI

```
XMLPI
 (
  [ NAME ] identifier
  [, value_expr ]
 )
```

XMLQUERY

```
XMLQUERY
 ( XQuery_string
   [ XML_passing_clause ]
   RETURNING CONTENT
 )
```

XMLROOT

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

XMLSEQUENCE

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

XMLSERIALIZE

```
XMLSERIALIZE
  ( { DOCUMENT | CONTENT } value_expr
      [ AS datatype ]
  )
```

XMLTABLE

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

XMLTABLE_options

```
[ XML_passing_clause ] [ COLUMNS XML_table_column
                                 [, XML_table_column
                                 ]...
                       ]
```

XMLTRANSFORM

```
XMLTRANSFORM(XMLType_instance, XMLType_instance)
```

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 datatype of its components.

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

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

CASE expression

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

Compound expression

```
{ (expr)
| { + | - | PRIOR } expr
| expr { * | / | + | - | || } expr
}
Note: The double vertical bars are part of the syntax
        (indicating concatenation) rather than BNF notation.
```

CURSOR expression

```
CURSOR (subquery)
```

DATETIME expression

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

Function expression

any built-in SQL function or user-defined function can be used as an expression

INTERVAL expression

```
interval_value_expr
   { DAY [ (leading_field_precision) ] TO
     SECOND [ (fractional_second_precision) ]
   | YEAR [ (leading_field_precision) ] TO
     MONTH
   }
```

Model expression

```
{ 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 expression

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

Scalar subquery expression

a subquery that returns exactly one column value from one row can be used as an expression

Simple expression

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

Type constructor expression

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

Variable expression

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

4 SQL Conditions

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

This chapter includes the following section:

- Syntax for SQL Condition Types

Syntax for SQL Condition Types

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

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

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

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 condition

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

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

IN conditions

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

IS A SET conditions

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

IS ANY condition

```
[ dimension_column IS ] ANY
```

IS EMPTY conditions

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

IS OF TYPE conditions

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

IS PRESENT condition

```
cell_reference IS PRESENT
```

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
```

Range conditions

```
expr [ NOT ] BETWEEN expr AND expr
```

REGEXP_LIKE condition

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

Simple comparison condition

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

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

SUBMULTISET conditions

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

UNDER_PATH condition

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

5 Subclauses

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

This chapter includes the following section:

- Syntax for Subclauses

Syntax for Subclauses

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

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

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

activate_standby_db_clause

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

add_binding_clause

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

add_column_clause

```
ADD
   ( column_definition [, column_definition] ... )
   [ column_properties ]
```

add_disk_clause

```
ADD
```

```
[ FAILGROUP failgroup_name ]
DISK qualified_disk_clause
     [, qualified_disk_clause ]...
  [ [ FAILGROUP failgroup_name ]
    DISK qualified_disk_clause
         [, qualified_disk_clause ]...
  ]...
```

add_hash_index_partition

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

add_hash_partition_clause

```
ADD PARTITION [ partition ]
   partitioning_storage_clause
   [ update_index_clauses ]
   [ parallel_clause ]
```

add_hash_subpartition

```
ADD subpartition_spec
   [ update_index_clauses ]
   [ parallel_clause ]
```

add_list_partition_clause

```
ADD PARTITION [ partition ]
   list_values_clause
   [ table_partition_description ]
   [ update_index_clauses ]
```

add_list_subpartition

```
ADD subpartition_spec
   [ update_index_clauses ]
```

add_logfile_clauses

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

add_overflow_clause

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

add_range_partition_clause

```
ADD PARTITION [ partition ]
   range_values_clause
   [ table_partition_description ]
   [ update_index_clauses ]
```

add_table_partition

```
{ add_range_partition_clause
| add_hash_partition_clause
| add_list_partition_clause
}
```

alias_file_name

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

allocate_extent_clause

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

alter_attribute_definition

```
{ { ADD | MODIFY } ATTRIBUTE
      { attribute [ datatype ]
      | ( attribute datatype
          [, attribute datatype ]...
        )
      }
| DROP ATTRIBUTE
     { attribute
     | ( attribute [, attribute ]... )
     }
}
```

alter_collection_clauses

```
MODIFY { LIMIT integer
       | ELEMENT TYPE datatype
       }
```

alter_datafile_clause

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

alter_external_table_clauses

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

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

alter_index_partitioning

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

alter_iot_clauses

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

alter_mapping_table_clauses

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

alter_method_spec

```
{ ADD | DROP }
{ map_order_function_spec
| subprogram_spec
}
  [ { ADD | DROP }
    { map_order_function_spec
    | subprogram_spec
    }
  ]...
```

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

```
{ OVERFLOW
     { allocate_extent_clause
     | shrink_clause
     | deallocate_unused_clause
     }
       [ allocate_extent_clause
       | shrink_clause
       | deallocate_unused_clause
       ]...
| add_overflow_clause
}
```

alter_session_set_clause

```
SET parameter_name = parameter_value
    [ parameter_name = parameter_value ]...
```

alter_system_reset_clause

```
parameter_name
   [ SCOPE = { MEMORY | SPFILE | BOTH } ]
   SID = 'sid'
```

alter_system_security_clauses

```
{ { ENABLE | DISABLE } RESTRICTED SESSION
| SET ENCRYPTION WALLET OPEN IDENTIFIED BY "password"
| SET ENCRYPTION WALLET CLOSE
| SET ENCRYPTION KEY [ "certificate_id" ] IDENTIFIED BY "password"
}
```

alter_system_set_clause

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

alter_table_partitioning

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

alter_table_properties

```
{ { physical_attributes_clause
  | logging_clause
  | table_compression
  | supplemental_table_logging
  | allocate_extent_clause
  | deallocate_unused_clause
  | shrink_clause
  | { CACHE | NOCACHE }
  | upgrade_table_clause
  | records_per_block_clause
  | parallel_clause
  | row_movement_clause
  }
    [ physical_attributes_clause
    | logging_clause
    | table_compression
    | supplemental_table_logging
    | allocate_extent_clause
    | deallocate_unused_clause
    | shrink_clause
    | { CACHE | NOCACHE }
    | upgrade_table_clause
    | records_per_block_clause
    | parallel_clause
    | row_movement_clause
    ]...
| RENAME TO new_table_name
}
[ alter_iot_clauses ]
```

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 )
```

analytic_clause

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

archive_log_clause

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

array_DML_clause

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

ASM_filename

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

attribute_clause

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

auditing_by_clause

```
BY { proxy [, proxy ]...
   | user [, user ]...
   }
```

auditing_on_clause

```
ON { [ schema. ]object
   | DIRECTORY directory_name
   | DEFAULT
   }
```

autoextend_clause

```
AUTOEXTEND
   { OFF
   | ON [ NEXT size_clause ]
        [ maxsize_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
```

build_clause

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

C_declaration

```
C [ NAME name ]
   LIBRARY lib_name
   [ AGENT IN (argument[, argument ]...) ]
   [ WITH CONTEXT ]
   [ PARAMETERS (parameter[, parameter ]...) ]
```

call_spec

```
LANGUAGE { Java_declaration | C_declaration }
```

cancel_clause

```
CANCEL
 { IMMEDIATE | WAIT | NOWAIT }
```

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_clauses

```
CHECK
{ ALL
| DISK
    disk_name
    [, disk_name ]...
| DISKS IN FAILGROUP
    failgroup_name
    [, failgroup_name ]...
| FILE
    filename
    [, filename ]...
}
[ REPAIR | NOREPAIR ]
```

checkpoint_clause

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

cluster_index_clause

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

coalesce_index_partition

```
COALESCE PARTITION
   [ parallel_clause ]
```

coalesce_table_partition

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

column_association

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

column_clauses

```
{ { add_column_clause
  | modify_column_clause
  | drop_column_clause
  }
    [ add_column_clause
    | modify_column_clause
    | drop_column_clause
    ]...
| rename_column_clause
| modify_collection_retrieval
  [ modify_collection_retrieval ]...
| modify_LOB_storage_clause
  [ modify_LOB_storage_clause ] ...
| alter_varray_col_properties
  [ alter_varray_col_properties ] ...
| REKEY encryption_spec
}
```

column_definition

```
  column datatype [ SORT ]
      [ DEFAULT expr ]
      [ ENCRYPT encryption_spec ]
      [ ( inline_constraint [ inline_constraint ] ... )
      | inline_ref_constraint
      ]
```

column_properties

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

commit_switchover_clause

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

compile_type_clause

```
COMPILE
   [ DEBUG ]
   [ SPECIFICATION | BODY ]
   [ compiler_parameters_clause
     [ compiler_parameters_clause ] ... ]
   [ REUSE SETTINGS ]
```

compiler_parameters_clause

```
parameter_name = parameter_value
```

composite_partitioning

```
PARTITION BY RANGE ( column_list )
  [ subpartition_by_list | subpartition_by_hash ]
  ( PARTITION [ partition ]
       range_values_clause
       table_partition_description
    [, PARTITION [ partition ]
          range_values_clause
          table_partition_description ] ...
  )
```

compute_statistics_clause

```
COMPUTE [ SYSTEM ] STATISTICS [ for_clause ]
```

conditional_insert_clause

```
[ ALL | FIRST ]
WHEN condition
THEN insert_into_clause
     [ values_clause ]
     [ error_logging_clause ]
     [ insert_into_clause
```

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

constraint

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

constraint_clauses

```
{ ADD { out_of_line_constraint
        [ out_of_line_constraint ]...
      | out_of_line_REF_constraint
      }
| MODIFY { CONSTRAINT constraint
         | PRIMARY KEY
         | UNIQUE (column [, column ]...)
         }
         constraint_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 ]
```

constructor_declaration

```
[ FINAL ]
[ INSTANTIABLE ]
CONSTRUCTOR FUNCTION datatype
[ [ SELF IN OUT datatype, ]
  parameter datatype
  [, parameter datatype ]...
]
RETURN SELF AS RESULT
```

```
{ IS | AS } { pl/sql_block | call_spec }
```

constructor_spec

```
[ FINAL ]
[ INSTANTIABLE ]
CONSTRUCTOR FUNCTION datatype
[ ([ SELF IN OUT datatype, ]
   parameter datatype
   [, parameter datatype ]...
  )
]
RETURN SELF AS RESULT
[ { IS | AS } call_spec ]
```

context_clause

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

controlfile_clauses

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

convert_standby_clause

```
CONVERT TO PHYSICAL STANDBY
```

cost_matrix_clause

```
COST MODEL
```

create_datafile_clause

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

create_incomplete_type

```
CREATE [ OR REPLACE ]
   TYPE [ schema. ]type_name ;
```

create_mv_refresh

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

create_nested_table_type

```
CREATE [ OR REPLACE ]
   TYPE [ schema. ]type_name
   [ OID 'object_identifier' ]
   { IS | AS } TABLE OF datatype ;
```

create_object_type

```
CREATE [ OR REPLACE ]
   TYPE [ schema. ]type_name
   [ OID 'object_identifier' ]
   [ invoker_rights_clause ]
   { { IS | AS } OBJECT
   | UNDER [schema.]supertype
   }
   [ sqlj_object_type ]
   [ ( attribute datatype
       [ sqlj_object_type_attr ]
       [, attribute datatype
          [ sqlj_object_type_attr ]...
       [, element_spec
          [, element_spec ]...
       ]
     )
   ]
   [ [ NOT ] FINAL ]
   [ [ NOT ] INSTANTIABLE ] ;
```

create_varray_type

```
CREATE [ OR REPLACE ]
   TYPE [ schema. ]type_name
   [ OID 'object_identifier' ]
   { IS | AS } { VARRAY | VARYING ARRAY }
   (limit) OF datatype ;
```

database_file_clauses

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

database_logging_clauses

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

datafile_tempfile_clauses

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

datafile_tempfile_spec

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

dblink

```
database[.domain [.domain ]... ]
[ @ connect_descriptor ]
```

dblink_authentication

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

db_user_proxy

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

deallocate_unused_clause

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

default_cost_clause

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

default_selectivity_clause

```
DEFAULT SELECTIVITY default_selectivity
```

default_tablespace

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

default_settings_clauses

```
{ SET DEFAULT
     { BIGFILE | SMALLFILE } TABLESPACE
| DEFAULT TABLESPACE tablespace
| DEFAULT TEMPORARY TABLESPACE
     { tablespace | tablespace_group_name }
| RENAME GLOBAL_NAME TO
     database.domain [.domain ]...
| { ENABLE BLOCK CHANGE TRACKING
    [ USING FILE 'filename' [ REUSE ] ]
  | DISABLE BLOCK CHANGE TRACKING
  }
| flashback_mode_clause
| set_time_zone_clause
}
```

default_temp_tablespace

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

dependent_handling_clause

```
{ INVALIDATE
| CASCADE [ { [ NOT ] INCLUDING TABLE DATA
            | CONVERT TO SUBSTITUTABLE
            }
          ]
     [ [FORCE ] exceptions_clause ]
}
```

dimension_join_clause

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

```
      }
     REFERENCES parent_level
  ]...
```

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_availability

```
{ MOUNT
| 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 | ALTER } TEMPLATE
      qualified_template_clause
      [, qualified_template_clause ]...
| DROP TEMPLATE
       template_name
       [, template_name ]...
}
```

distributed_recov_clauses

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

dml_event_clause

```
{ DELETE | INSERT | UPDATE
     [ OF column [, column ]... ]
}
  [ OR { DELETE | INSERT | UPDATE
       [ OF column [, column]... ]
       }
  ]...
ON { [ schema. ]table
   | [ NESTED TABLE nested_table_column OF ]
          [ schema. ] view
   }
[ referencing_clause ]
[ FOR EACH ROW ]
```

dml_table_expression_clause

```
{ [ schema. ]
  { table
    [ { PARTITION (partition)
      | SUBPARTITION (subpartition)
      }
    | @ dblink
    ]
  | { view | materialized view } [ @ dblink ]
  }
| ( subquery [ subquery_restriction_clause ] )
| table_collection_expression
}
```

domain_index_clause

```
INDEXTYPE IS indextype
   [ 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 }
      [ CASCADE CONSTRAINTS | INVALIDATE ]...
  ]
| DROP { COLUMN column
       | (column [, column ]...)
       }
  [ { CASCADE CONSTRAINTS | INVALIDATE }
      [ 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
     [ CASCADE ]
   }
```

drop_disk_clauses

```
DROP
{ DISK
    disk_name [ FORCE | NOFORCE ]
    [, disk_name [ FORCE | NOFORCE ] ]...
| DISKS IN FAILGROUP
```

```
    failgroup_name [ FORCE | NOFORCE ]
    [, failgroup_name [ FORCE | NOFORCE ] ]...
}
```

drop_diskgroup_file_clause

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

drop_index_partition

```
DROP PARTITION partition_name
```

drop_logfile_clauses

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

drop_table_partition

```
DROP PARTITION partition
   [ update_index_clauses [ parallel_clause ] ]
```

drop_table_subpartition

```
DROP SUBPARTITION subpartition
   [ update_index_clauses [ parallel_clause ] ]
```

element_spec

```
[ inheritance_clauses ]
{ subprogram_spec
| constructor_spec
| map_order_function_spec
}
  [ subprogram_clause
  | constructor_spec
  | map_order_function_spec
  ]...
[, pragma_clause ]
```

else_clause

```
ELSE else_expr
```

enable_disable_clause

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

end_session_clauses

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

encryption_spec

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

error_logging_clause

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

estimate_statistics_clause

```
ESTIMATE [ SYSTEM ] STATISTICS [ for_clause ]
[ SAMPLE integer { ROWS | PERCENT } ]
```

exceptions_clause

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

exchange_partition_subpart

```
EXCHANGE { PARTITION partition
         | SUBPARTITION subpartition
         }
   WITH TABLE table
   [ { INCLUDING | EXCLUDING } INDEXES ]
   [ { WITH | WITHOUT } VALIDATION ]
   [ exceptions_clause ]
   [ update_index_clauses [ parallel_clause ] ]
```

expr

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

expression_list

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

extended_attribute_clause

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

extent_management_clause

```
EXTENT MANAGEMENT
   { LOCAL
     [ AUTOALLOCATE
     | UNIFORM
       [ SIZE size_clause ]
     ]
   | DICTIONARY
   }
```

external_data_properties

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

external_table_clause

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

file_specification

```
{ datafile_tempfile_spec
| redo_log_file_spec
}
```

finish_clause

```
 FINISH [ FORCE ] [ WAIT | NOWAIT ]
```

flashback_mode_clause

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

flashback_query_clause

```
[ VERSIONS BETWEEN
  { SCN | TIMESTAMP }
  { expr | MINVALUE } AND
  { expr | MAXVALUE }
]
AS OF { SCN | TIMESTAMP } expr
```

for_clause

```
FOR
   { TABLE
   | ALL [ INDEXED ] COLUMNS [ SIZE integer ]
   | COLUMNS [ SIZE integer ]
     { column | attribute } [ SIZE integer ]
       [ { column | attribute }
         [ SIZE integer ]
       ]...
   | ALL [ LOCAL ] INDEXES
   }
   [ FOR
    { TABLE
    | ALL [ INDEXED ] COLUMNS
         [ SIZE integer ]
    | COLUMNS [ SIZE integer ]
      { column | attribute } [ SIZE integer ]
        [ { column | attribute }
          [ SIZE integer ]
        ]...
    | ALL [ LOCAL ] INDEXES
    }
   ]...
```

for_update_clause

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

full_database_recovery

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

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 ]
  }
}
```

function_declaration

```
FUNCTION name
   (parameter datatype[, parameter datatype ]...)
   RETURN datatype
   { IS | AS } { pl/sql_block | call_spec }
```

function_spec

```
FUNCTION name
   (parameter datatype [, parameter datatype ]...)
   return_clause
```

general_recovery

```
RECOVER
[ AUTOMATIC ]
[ FROM 'location' ]
{ { full_database_recovery
  | partial_database_recovery
  | LOGFILE 'filename'
  }
  [ { TEST
    | ALLOW integer CORRUPTION
    | NOPARALLEL
    }
      [ TEST
      | ALLOW integer CORRUPTION
      | NOPARALLEL
      ]...
  ]
| 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_system_privileges

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

grantee_clause

```
{ user [ IDENTIFIED BY password ]
| role
| PUBLIC
}
  [, { user [ IDENTIFIED BY password ]
     | role
     | PUBLIC
     }
  ]...
```

group_by_clause

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

grouping_expression_list

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

grouping_sets_clause

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

hash_partitioning

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

hash_partitions_by_quantity

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

hierarchical_query_clause

```
[ START WITH condition ]
```

```
CONNECT BY [ NOCYCLE ] condition
```

hierarchy_clause

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

implementation_clause

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

incomplete_file_name

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

index_attributes

```
[ { physical_attributes_clause
  | logging_clause
  | ONLINE
  | COMPUTE STATISTICS
  | TABLESPACE { tablespace | DEFAULT }
  | key_compression
  | { SORT | NOSORT }
  | REVERSE
  | parallel_clause
  }
    [ physical_attributes_clause
    | logging_clause
    | ONLINE
    | COMPUTE STATISTICS
    | TABLESPACE { tablespace | DEFAULT }
    | key_compression
    | { SORT | NOSORT }
    | REVERSE
    | parallel_clause
    ]...
]
```

index_expr

```
{ column | column_expression }
```

index_org_overflow_clause

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

index_org_table_clause

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

index_partition_description

```
PARTITION
[ partition
   [ { segment_attributes_clause
     | key_compression
     }
     [ segment_attributes_clause
     | key_compression
     ]...
   ]
]
```

index_partitioning_clause

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

index_properties

```
[ { { global_partitioned_index
    | local_partitioned_index
    }
  | index_attributes
  }
      [ { { global_partitioned_index
          | local_partitioned_index
          }
        | index_attributes
        }
      ]...
| domain_index_clause
]
```

index_subpartition_clause

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

individual_hash_partitions

```
(PARTITION
    [ partition partitioning_storage_clause ]
  [, PARTITION
        [ partition partitioning_storage_clause ]
  ]...
)
```

inheritance_clauses

```
[ NOT ] { OVERRIDING | FINAL | INSTANTIABLE }
  [ [ NOT ] { OVERRIDING | FINAL | INSTANTIABLE } ]...
```

inline_constraint

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

inline_ref_constraint

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

inner_cross_join_clause

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

insert_into_clause

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

instance_clauses

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

integer

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

interval_day_to_second

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

interval_year_to_month

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

into_clause

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

invoker_rights_clause

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

Java_declaration

```
JAVA NAME string
```

join_clause

```
table_reference { inner_cross_join_clause | outer_join_clause
                  [ inner_cross_join_clause | outer_join_clause ]...
                }
```

key_compression

```
{ COMPRESS [ integer ]
| NOCOMPRESS
}
```

level_clause

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

list_partitioning

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

list_values_clause

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

LOB_parameters

```
{ TABLESPACE tablespace
| { ENABLE | DISABLE } STORAGE IN ROW
| storage_clause
| CHUNK integer
| PCTVERSION integer
| RETENTION
| FREEPOOLS integer
| { CACHE
  | { NOCACHE | CACHE READS } [ logging_clause ]
  }
}
  [ TABLESPACE tablespace
  | { ENABLE | DISABLE } STORAGE IN ROW
  | storage_clause
  | CHUNK integer
  | PCTVERSION integer
  | RETENTION
  | FREEPOOLS integer
  | { CACHE
    | { NOCACHE | CACHE READS } [ logging_clause ]
    }
  ]...
```

LOB_partition_storage

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

LOB_storage_clause

```
LOB
{ (LOB_item [, LOB_item ]...)
     STORE AS (LOB_parameters)
| (LOB_item)
     STORE AS
        { LOB_segname (LOB_parameters)
        | LOB_segname
        | (LOB_parameters)
        }
}
```

local_partitioned_index

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

logfile_clause

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

logfile_clauses

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

logfile_descriptor

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

logging_clause

```
{ LOGGING | NOLOGGING }
```

main_model

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

managed_standby_recovery

```
RECOVER
 { MANAGED STANDBY DATABASE
   [ { redo_apply_clauses | finish_clause | cancel_clause } ]
 |
   TO LOGICAL STANDBY db_name
 }
```

map_order_func_declaration

```
{ MAP | ORDER } MEMBER function_declaration
```

map_order_function_spec

```
{ MAP | ORDER } MEMBER function_spec
```

mapping_table_clauses

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

materialized_view_props

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

maximize_standby_db_clause

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

maxsize_clause

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

merge_insert_clause

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

merge_table_partitions

```
MERGE PARTITIONS partition_1, partition_2
```

```
   [ INTO partition_spec ]
   [ update_index_clauses ]
   [ parallel_clause ]
```

merge_table_subpartitions

```
MERGE SUBPARTITIONS subpart_1, subpart_2
   [ INTO subpartition_spec ]
   [ update_index_clauses ]
   [ parallel_clause ]
```

merge_update_clause

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

mining_attribute_clause

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

model_clause

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

model_column

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

model_column_clauses

```
[ query_partition_clause [ c_alias ] ]
DIMENSION BY (model_column
               [, model_column ]...)
MEASURES (model_column
           [, model_column ]...)
```

model_rules_clause

```
RULES
[ { UPDATE | UPSERT [ ALL ] } ]
[ { AUTOMATIC | SEQUENTIAL } ORDER ]
[ ITERATE (number) [ UNTIL (condition) ] ]
([ { UPDATE | UPSERT [ ALL ] } ]
 cell_assignment [ order_by_clause ] = expr
   [ [ { UPDATE | UPSERT [ ALL ] } ]
     cell_assignment [ order_by_clause ] = expr
   ]...
)
```

modify_col_properties

```
( column [ datatype ]
         [ DEFAULT expr ]
         [ { ENCRYPT encryption_spec | DECRYPT }
         [ inline_constraint
           [ inline_constraint ]... ]
         [ LOB_storage_clause ]
  [, column [ datatype ]
            [ DEFAULT expr ]
            [ { ENCRYPT encryption_spec | DECRYPT }
            [ inline_constraint
              [ inline_constraint ]... ]
            [ LOB_storage_clause ]
  ]
)
```

modify_col_substitutable

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

modify_collection_retrieval

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

modify_column_clauses

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

modify_hash_partition

```
MODIFY PARTITION partition
  { partition_attributes
  | alter_mapping_table_clause
  | [ REBUILD ] UNUSABLE LOCAL INDEXES
  }
```

modify_hash_subpartition

```
{ { allocate_extent_clause
  | deallocate_unused_clause
  | shrink_clause
  | { LOB LOB_item
    | VARRAY varray
    }
    modify_LOB_parameters
      [ { LOB LOB_item
        | VARRAY varray
        }
        modify_LOB_parameters
      ]...
  }
| [ REBUILD ] UNUSABLE LOCAL INDEXES
}
```

modify_index_default_attrs

```
MODIFY DEFAULT ATTRIBUTES
   [ FOR PARTITION partition ]
   { physical_attributes_clause
   | TABLESPACE { tablespace | DEFAULT }
   | logging_clause
   }
     [ 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
  | key_compression
  }
    [ deallocate_unused_clause
    | allocate_extent_clause
    | physical_attributes_clause
    | logging_clause
    | key_compression
    ]...
| PARAMETERS ('ODCI_parameters')
| COALESCE
| UPDATE BLOCK REFERENCES
| UNUSABLE
}
```

modify_index_subpartition

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

modify_list_partition

```
MODIFY PARTITION partition
  { partition_attributes
  | { ADD | DROP } VALUES
    (literal[, literal ]...)
  | [ REBUILD ] UNUSABLE LOCAL INDEXES
  }
```

modify_list_subpartition

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

modify_LOB_parameters

```
{ storage_clause
| PCTVERSION integer
| RETENTION
| FREEPOOLS integer
| REBUILD FREEPOOLS
| { CACHE
```

```
  | { NOCACHE | CACHE READS } [ logging_clause ]
  }
| allocate_extent_clause
| shrink_clause
| deallocate_unused_clause
}
  [ storage_clause
  | PCTVERSION integer
  | RETENTION
  | FREEPOOLS integer
  | REBUILD FREEPOOLS
  | { 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_range_partition

```
MODIFY PARTITION partition
   { partition_attributes
   | { add_hash_subpartition
     | add_list_subpartition
     }
   | COALESCE SUBPARTITION
        [ update_index_clauses ]
        [ parallel_clause ]
   | alter_mapping_table_clause
   | [ REBUILD ] UNUSABLE LOCAL INDEXES
   }
```

modify_table_default_attrs

```
MODIFY DEFAULT ATTRIBUTES
   [ FOR PARTITION partition ]
   [ segment_attributes_clause ]
   [ table_compression ]
   [ PCTTHRESHOLD integer ]
   [ key_compression ]
   [ alter_overflow_clause ]
   [ { LOB (LOB_item)
     | VARRAY varray
     }
     (LOB_parameters)
     [ { LOB (LOB_item)
       | VARRAY varray
       }
       (LOB_parameters)
     ]...
   ]
```

modify_table_partition

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

modify_table_subpartition

```
MODIFY SUBPARTITION subpartition
{ modify_hash_subpartition
| modify_list_subpartition
}
```

move_table_clause

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

move_table_partition

```
MOVE PARTITION partition
   [ MAPPING TABLE ]
   [ table_partition_description ]
   [ update_index_clauses ]
   [ parallel_clause ]
```

move_table_subpartition

```
MOVE SUBPARTITION
   subpartition_spec
   [ update_index_clauses ]
   [ parallel_clause ]
```

multi_column_for_loop

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

multi_table_insert

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

multiset_except

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

multiset_intersect

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

multiset_union

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

nested_table_col_properties

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

new_values_clause

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

number

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

numeric_file_name

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

object_properties

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

object_table

```
CREATE [ GLOBAL TEMPORARY ] TABLE
   [ schema. ]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
  { DEFAULT | ( attribute
                [, attribute ]... )
  }
| UNDER [ schema. ]superview
}
({ out_of_line_constraint
 | attribute inline_constraint
             [ inline_constraint ]...
 }
   [, { out_of_line_constraint
      | attribute inline_constraint
                  [ inline_constraint ]...
      }
   ]...
)
```

OID_clause

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

OID_index_clause

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

on_comp_partitioned_table

```
[ STORE IN ( tablespace [, tablespace ]... ) ]
( PARTITION
    [ partition
      [ { segment_attribute_clause
        | key_compression
        }
          [ segment_attribute_clause
          | key_compression
          ]...
      ]
      [ index_subpartition_clause ]
    ]
      [, PARTITION
           [ partition
             [ { segment_attribute_clause
               | key_compression
               }
```

```
                [ segment_attribute_clause
                | key_compression
                ]...
            ]
            [ index_subpartition_clause ]
          ]...
     ]
)
```

on_hash_partitioned_table

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

on_list_partitioned_table

```
( PARTITION
    [ partition
      [ { segment_attributes_clause
        | key_compression
        }
          [ segment_attributes_clause
          | key_compression
          ]...
      ]
    ]
  [, PARTITION
       [ partition
         [ { segment_attributes_clause
           | key_compression
           }
             [ segment_attributes_clause
             | key_compression
             ]...
         ]
       ]
  ]...
)
```

on_object_clause

```
{ schema.object
| { DIRECTORY directory_name
  | JAVA { SOURCE | RESOURCE } [ schema. ]object
  }
}
```

on_range_partitioned_table

```
( PARTITION
    [ partition
      [ { segment_attributes_clause
        | key_compression
        }
          [ segment_attributes_clause
          | key_compression
          ]...
      ]
    ]
  [, PARTITION
       [ partition
```

```
        [ { segment_attributes_clause
          | key_compression
          }
            [ segment_attributes_clause
            | key_compression
            ]...
        ]
      ]
  ]...
)
```

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 ]
  ]...
```

out_of_line_constraint

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

out_of_line_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_attr })
  references_clause
  [ constraint_state ]
}
```

outer_join_clause

```
[ query_partition_clause ]
{ outer_join_type JOIN
| 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_enable_clause

```
PARALLEL_ENABLE
[ (PARTITION argument BY
      { ANY
      | { HASH | RANGE } (column [, column ]...)
      }
  )
 [ streaming_clause ]
]
```

partial_database_recovery

```
{ TABLESPACE tablespace [, tablespace ]...
| DATAFILE { 'filename' | filenumber }
             [, 'filename' | filenumber ]...
           }
| STANDBY
  { TABLESPACE tablespace [, tablespace ]...
  | DATAFILE { 'filename' | filenumber }
               [, 'filename' | filenumber ]...
             }
  }
  UNTIL [ CONSISTENT WITH ] CONTROLFILE
}
```

partition_attributes

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

partition_extended_name

```
[ schema.] { table | view }
[ PARTITION (partition)
| SUBPARTITION (subpartition)
]
```

partition_level_subpartition

```
{ SUBPARTITIONS hash_subpartition_quantity
  [ STORE IN (tablespace[, tablespace ]...) ]
| (subpartition_spec[, subpartition_spec ]...)
}
```

partition_spec

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

partitioning_storage_clause

```
[ { TABLESPACE tablespace
  | OVERFLOW [ TABLESPACE tablespace ]
  | LOB (LOB_item) STORE AS
    { LOB_segname [ (TABLESPACE tablespace) ]
    | (TABLESPACE tablespace)
    }
  | VARRAY varray_item STORE AS LOB LOB_segname
  }
    [ { TABLESPACE tablespace
      | OVERFLOW [ TABLESPACE tablespace ]
      | LOB (LOB_item) STORE AS
        { LOB_segname [ (TABLESPACE tablespace) ]
        | (TABLESPACE tablespace)
        }
      | VARRAY varray_item STORE AS LOB LOB_segname
      }
    ]...
]
```

password_parameters

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

permanent_tablespace_clause

```
{ MINIMUM EXTENT size_clause
| BLOCKSIZE integer [ K ]
| logging_clause
| FORCE LOGGING
| DEFAULT [ table_compression ]
  storage_clause
| { ONLINE | OFFLINE }
| extent_management_clause
| segment_management_clause
| flashback_mode_clause
  [ MINIMUM EXTENT size_clause
  | BLOCKSIZE integer [ K ]
  | logging_clause
  | FORCE LOGGING
  | DEFAULT [ table_compression ]
    storage_clause
  | { ONLINE | OFFLINE }
  | extent_management_clause
  | segment_management_clause
  | flashback_mode_clause
  ]...
}
```

physical_attributes_clause

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

physical_properties

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

pragma_clause

```
PRAGMA RESTRICT_REFERENCES
({ method_name | DEFAULT } ,
 { RNDS | WNDS | RNPS | WNPS | TRUST }
   [, { RNDS | WNDS | RNPS | WNPS | TRUST } ]...
)
```

procedure_declaration

```
PROCEDURE name (parameter datatype
                [, parameter datatype ]...)
   { IS | AS } { pl/sql_block | call_spec }
```

procedure_spec

```
PROCEDURE name
(parameter datatype [, parameter datatype ]...)
[ { IS | AS } call_spec ]
```

proxy_clause

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

qualified_disk_clause

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

qualified_template_clause

```
template_name
ATTRIBUTES
([ MIRROR | UNPROTECTED ]
 [ FINE | COARSE ]
)
```

query_partition_clause

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

query_table_expression

```
{ query_name
| [ schema. ]
  { table [ { PARTITION (partition)
            | SUBPARTITION (subpartition)
            }
            [ sample_clause ]
          | [ sample_clause ]
          | @ dblink
          ]
  | { view | materialized view } [ @ dblink ]
  }
| (subquery [ subquery_restriction_clause ])
| table_collection_expression
}
```

quiesce_clauses

```
QUIESCE RESTRICTED | UNQUIESCE
```

range_partitioning

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

range_values_clause

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

rebalance_diskgroup_clause

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

rebuild_clause

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

records_per_block_clause

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

recovery_clauses

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

redo_apply_clauses

```
 { USING CURRENT LOGFILE
 | NOPARALLEL
 | DISCONNECT [ FROM SESSION ]
 | NODELAY
 | UNTIL CHANGE integer
 }
  [ { USING CURRENT LOGFILE
    | NOPARALLEL
    | DISCONNECT [ FROM SESSION ]
    | NODELAY
    | UNTIL CHANGE integer
    } ]...
```

redo_log_file_spec

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

reference_model

```
REFERENCE reference_spreadsheet_name
ON (subquery)
spreadsheet_column_clauses
[ cell_reference_options ]
```

references_clause

```
REFERENCES [ schema. ] { object_table | view }
```

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

referencing_clause

```
REFERENCING
 { OLD [ AS ] old
 | NEW [ AS ] new
 | PARENT [ AS ] parent }
   [ OLD [ AS ] old
   | NEW [ AS ] new
   | PARENT [ AS ] parent ]...
```

register_logfile_clause

```
REGISTER
[ OR REPLACE ]
[ PHYSICAL | LOGICAL ]
LOGFILE
[ file_specification
  [, file_specification ]... ]
[ FOR logminer_session_name ]
```

relational_properties

```
{ column_definition
| { out_of_line_constraint
  | out_of_line_ref_constraint
  | supplemental_logging_props
  }
}
  [, { column_definition
     | { out_of_line_constraint
       | out_of_line_ref_constraint
       | supplemental_logging_props
       }
  ]...
```

relational_table

```
CREATE [ GLOBAL TEMPORARY ] TABLE [ schema. ]table
   [ (relational_properties) ]
   [ ON COMMIT { DELETE | PRESERVE } ROWS ]
   [ physical_properties ]
   [ table_properties ] ;
```

rename_column_clause

```
RENAME COLUMN old_name TO new_name
```

rename_index_partition

```
RENAME { PARTITION partition
       | SUBPARTITION subpartition }
   TO new_name
```

rename_partition_subpart

```
RENAME { PARTITION | SUBPARTITION }
  current_name TO new_name
```

replace_type_clause

```
REPLACE [ invoker_rights_clause ] AS OBJECT
   (attribute datatype [, attribute datatype ]...
   [, element_spec [, element_spec ]... ])
```

resize_disk_clauses

```
RESIZE
{ ALL [ SIZE size_clause ]
| DISK
   disk_name [ SIZE size_clause ]
   [, disk_name [ SIZE size_clause ] ]...
| DISKS IN FAILGROUP
    failgroup_name [ SIZE size_clause ]
    [, failgroup_name [ SIZE size_clause ] ]...
}
```

resource_parameters

```
{ { SESSIONS_PER_USER
  | CPU_PER_SESSION
  | CPU_PER_CALL
  | CONNECT_TIME
  | IDLE_TIME
  | LOGICAL_READS_PER_SESSION
  | LOGICAL_READS_PER_CALL
  | COMPOSITE_LIMIT
  }
  { integer | UNLIMITED | DEFAULT }
| PRIVATE_SGA
  { size_clause | UNLIMITED | DEFAULT }
}
```

return_clause

```
{ RETURN datatype [ { IS | AS } call_spec ]
| sqlj_object_type_sig
}
```

return_rows_clause

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

returning_clause

```
RETURNING expr [, expr ]...
INTO data_item [, data_item ]...
```

revoke_object_privileges

```
{ object_privilege | ALL [ PRIVILEGES ] }
  [, { object_privilege | ALL [ PRIVILEGES ] } ]...
on_object_clause
FROM grantee_clause
[ CASCADE CONSTRAINTS | FORCE ]
```

revoke_system_privileges

```
{ system_privilege
| role
| ALL PRIVILEGES
}
  [, { system_privilege
     | role
     | ALL PRIVILEGES
     }
  ]...
FROM grantee_clause
```

rollup_cube_clause

```
{ ROLLUP | CUBE } (grouping_expression_list)
```

routine_clause

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

row_movement_clause

```
{ ENABLE | DISABLE } ROW MOVEMENT
```

sample_clause

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

schema_object_clause

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

scoped_table_ref_constraint

```
{ SCOPE FOR
  ({ ref_column | ref_attribute })
  IS [ schema. ] { scope_table_name | c_alias }
}
  [, SCOPE FOR
     ({ ref_column | ref_attribute })
     IS [ schema. ] { scope_table_name | c_alias }
  ]...
```

searched_case_expression

```
WHEN condition THEN return_expr
[ WHEN condition THEN return_expr ]...
```

security_clause

```
GUARD { ALL | STANDBY | NONE }
```

segment_attributes_clause

```
{ physical_attributes_clause
| TABLESPACE tablespace
| logging_clause
}
  [ physical_attributes_clause
  | TABLESPACE tablespace
  | logging_clause
  ]...
```

segment_management_clause

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

select_list

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

set_subpartition_template

```
SET SUBPARTITION TEMPLATE
   { (SUBPARTITION subpartition
         [ list_values_clause ]
         [ partitioning_storage_clause ]
       [, SUBPARTITION subpartition
             [ list_values_clause ]
             [ partitioning_storage_clause ]...
       ]
      )
   | hash_subpartition_quantity
   }
```

set_time_zone_clause

```
SET TIME_ZONE =
   '{ { + | - } hh : mi | time_zone_region }'
```

shrink_clause

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

shutdown_dispatcher_clause

```
SHUTDOWN [ IMMEDIATE ] dispatcher_name
```

simple_case_expression

```
expr WHEN comparison_expr
     THEN return_expr
     [ WHEN comparison_expr
       THEN return_expr ]...
```

single_column_for_loop

```
FOR dimension_column
  { IN ( { literal
           [, literal ]...
         | subquery
         }
       )
  | [ LIKE pattern ]
    FROM literal TO literal
      { INCREMENT | DECREMENT } literal
  }
```

single_table_insert

```
insert_into_clause
{ values_clause [ returning_clause ]
| subquery
}
[ error_logging_clause ]
```

size_clause

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

split_index_partition

```
SPLIT PARTITION partition_name_old
```

```
   AT (literal [, literal ]...)
   [ INTO (index_partition_description,
           index_partition_description
          )
   ]
   [ parallel_clause ]
```

split_table_partition

```
SPLIT PARTITION current_partition
   { AT | VALUES } (literal [, literal ]...)
   [ INTO (partition_spec, partition_spec) ]
   [ update_index_clauses ]
   [ parallel_clause ]
```

split_table_subpartition

```
SPLIT SUBPARTITION subpartition
   VALUES ({ literal | NULL }
             [, literal | NULL ]...)
   [ INTO (subpartition_spec,
           subpartition_spec
          )
   ]
   [ update_index_clauses ]
   [ parallel_clause ]
```

sql_statement_clause

```
{ { statement_option | ALL }
  [, { statement_option | ALL } ]...
| { system_privilege | ALL PRIVILEGES }
  [, { system_privilege | ALL PRIVILEGES } ]...
}
[ auditing_by_clause ]
```

sqlj_object_type

```
EXTERNAL NAME java_ext_name LANGUAGE JAVA
   USING (SQLData | CustomDatum | OraData)
```

sqlj_object_type_attr

```
EXTERNAL NAME 'field_name'
```

sqlj_object_type_sig

```
RETURN { datatype | SELF AS RESULT }
EXTERNAL { VARIABLE NAME 'java_static_field_name'
         | NAME 'java_method_sig'
         }
```

standby_database_clauses

```
( activate_standby_db_clause
| maximize_standby_db_clause
| register_logfile_clause
| commit_switchover_clause
| start_standby_clause
| stop_standby_clause
| convert_standby_clause
)
[ parallel_clause ]
```

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
       }
}
```

stop_standby_clause

```
{ STOP | ABORT }
LOGICAL STANDBY APPLY
```

storage_clause

```
STORAGE
   ({ INITIAL size_clause
    | NEXT size_clause
    | MINEXTENTS integer
    | MAXEXTENTS { integer | UNLIMITED }
    | PCTINCREASE integer
    | FREELISTS integer
    | FREELIST GROUPS integer
    | OPTIMAL [ size_clause
              | NULL
              ]
    | BUFFER_POOL { KEEP | RECYCLE | DEFAULT }
    }
      [ INITIAL size_clause
      | NEXT size_clause
      | MINEXTENTS integer
      | MAXEXTENTS { integer | UNLIMITED }
      | PCTINCREASE integer
      | FREELISTS integer
      | FREELIST GROUPS integer
      | OPTIMAL [ size_clause
                | NULL
                ]
      | BUFFER_POOL { KEEP | RECYCLE | DEFAULT }
      ]...
   )
```

streaming_clause

```
{ ORDER | CLUSTER } BY (column [, column ]...)
```

subpartition_by_hash

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

subpartition_by_list

```
SUBPARTITION BY LIST (column)
   [ subpartition_template ]
```

subpartition_spec

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

subpartition_template

```
SUBPARTITION TEMPLATE
   (SUBPARTITION subpartition
       [ list_values_clause ]
       [ partitioning_storage_clause ]
     [, SUBPARTITION subpartition
           [ list_values_clause ]
           [ partitioning_storage_clause ]
     ]
   )
   | hash_subpartition_quantity
```

subprogram_declaration

```
{ MEMBER | STATIC }
   { procedure_declaration
   | function_declaration
   | constructor_declaration
   }
```

subprogram_spec

```
{ MEMBER | STATIC }
{ procedure_spec | function_spec }
```

subquery

```
[ subquery_factoring_clause ]
SELECT
   [ hint ]
   [ { { DISTINCT | UNIQUE }
     | ALL
     }
   ]
   select_list
   FROM { table_reference [, table_reference ]...
               | join_clause
               | ( join_clause )
               }
   [ where_clause ]
   [ hierarchical_query_clause ]
   [ group_by_clause ]
   [ HAVING condition ]
   [ model_clause ]
   [ { UNION [ ALL ]
     | INTERSECT
     | MINUS
     }
     (subquery)
   ]
   [ order_by_clause ]
```

subquery_factoring_clause

```
WITH query_name AS (subquery)
```

```
     [, query_name AS (subquery) ]...
```

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_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_grp_clause
| supplemental_id_key_clause
}
```

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
            }
       ]...
}
```

table_collection_expression

```
TABLE (collection_expression) [ (+) ]
```

table_compression

```
{ COMPRESS | NOCOMPRESS }
```

table_index_clause

```
[ schema. ]table [ t_alias ]
(index_expr [ ASC | DESC ]
  [, index_expr [ ASC | DESC ] ]...)
[ index_properties ]
```

table_partition_description

```
[ segment_attributes_clause ]
[ table_compression | key_compression ]
[ OVERFLOW [ segment_attributes_clause ] ]
[ { LOB_storage_clause
  | varray_col_properties
  }
    [ LOB_storage_clause
    | varray_col_properties
    ]...
]
[ partition_level_subpartition ]
```

table_partitioning_clauses

```
{ range_partitioning
| hash_partitioning
| list_partitioning
| composite_partitioning
}
```

table_properties

```
[ column_properties ]
[ table_partitioning_clauses ]
[ CACHE | NOCACHE ]
[ parallel_clause ]
[ ROWDEPENDENCIES | NOROWDEPENDENCIES ]
[ enable_disable_clause ]
  [ enable_disable_clause ]...
[ row_movement_clause ]
[ AS subquery ]
```

table_reference

```
{ ONLY
  (query_table_expression)
  [ flashback_query_clause ]
  [ t_alias ]
| query_table_expression
  [ flashback_query_clause ]
  [ t_alias ]
}
```

tablespace_clauses

```
{ EXTENT MANAGEMENT LOCAL
| DATAFILE file_specification
           [, file_specification ]...
| SYSAUX DATAFILE file_specification
                  [, file_specification ]...
| default_tablespace
| default_temp_tablespace
| undo_tablespace
}
```

tablespace_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 }
```

temporary_tablespace_clause

```
TEMPORARY TABLESPACE tablespace
  [ TEMPFILE file_specification
             [, file_specification ]...
  ]
  [ tablespace_group_clause ]
  [ extent_management_clause ]
```

text

```
[ {N | n} ]
{ 'c [ c ]...'
| { Q | q }
  'quote_delimiter c [ c ]... quote_delimiter'
}
```

trace_file_clause

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

truncate_partition_subpart

```
TRUNCATE { PARTITION partition
         | SUBPARTITION subpartition
         }
   [ { DROP | REUSE } STORAGE ]
   [ update_index_clauses [ parallel_clause ] ]
```

undo_tablespace

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

undo_tablespace_clause

```
UNDO TABLESPACE tablespace
  [ DATAFILE file_specification
             [, file_specification ]...
  ]
  [ extent_management_clause ]
  [ tablespace_retention_clause ]
```

undrop_disk_clause

```
UNDROP DISKS
```

update_all_indexes_clause

```
UPDATE INDEXES
   [ (index ( { update_index_partition
              | update_index_subpartition
              }
            )
     )
       [, (index ( { update_index_partition
                   | update_index_subparition
                   }
                 )
          )
       ]...
```

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 ]
```

using_function_clause

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

using_index_clause

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

using_statistics_type

```
USING { [ schema. ] statistics_type | NULL }
```

using_type_clause

```
USING [ schema. ]implementation_type
[ array_DML_clause ]
```

validation_clauses

```
{ VALIDATE REF UPDATE
     [ SET DANGLING TO NULL ]
| VALIDATE STRUCTURE
     [ CASCADE ]
     [ into_clause ]
     { OFFLINE| ONLINE }
}
```

values_clause

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

varray_col_properties

```
VARRAY varray_item
   { [ substitutable_column_clause ]
     STORE AS LOB
        { [ LOB_segname ] (LOB_parameters)
        | LOB_segname
        }
   | substitutable_column_clause
   }
```

where_clause

```
WHERE condition
```

windowing_clause

```
{ ROWS | RANGE }
{ BETWEEN
  { UNBOUNDED PRECEDING
  | CURRENT ROW
  | value_expr { PRECEDING | FOLLOWING }
  }
  AND
  { UNBOUNDED FOLLOWING
  | CURRENT ROW
  | value_expr { PRECEDING | FOLLOWING }
  }
| { UNBOUNDED PRECEDING
  | CURRENT ROW
  | value_expr PRECEDING
  }
}
```

XML_attributes_clause

```
XMLATTRIBUTES
  (value_expr [ AS c_alias ]
    [, value_expr [ AS c_alias ]
      ]...
  )
```

XML_namespaces_clause

```
XMLNAMESPACES
  ( [ string AS identifier ]
      [ [, string AS identifier ]
      ]...
    [ DEFAULT string ]
  )
```

XML_passing_clause

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

XML_table_column

```
column
     { FOR ORDINALITY
     | datatype [ PATH string ] [ DEFAULT expr ]
     }
```

XMLSchema_spec

```
[ XMLSCHEMA XMLSchema_URL ]
ELEMENT { element | XMLSchema_URL # element }
```

XMLType_column_properties

```
XMLTYPE [ COLUMN ] column
   [ XMLType_storage ]
   [ XMLSchema_spec ]
```

XMLType_storage

```
STORE AS
   { OBJECT RELATIONAL
   | CLOB [ { LOB_segname [ (LOB_parameters) ]
            | LOB_parameters
            }
          ]
```

XMLType_table

```
CREATE TABLE [ GLOBAL TEMPORARY ] TABLE
  [ schema. ]table OF XMLTYPE
  [ (oject_properties) ]
  [ XMLTYPE XMLType_storage ]
  [ XMLSchema_spec ]
  [ ON COMMIT { DELETE | PRESERVE } ROWS ]
  [ OID_clause ]
  [ OID_index_clause ]
  [ physical_properties ]
  [ table_properties ] ;
```

XMLType_view_clause

```
OF XMLTYPE
[ XMLSchema_spec ]
WITH OBJECT IDENTIFIER
{ DEFAULT | ( expr [, expr ]...) }
```

6 Datatypes

This chapter presents datatypes that are recognized by Oracle and available for use within SQL.

This chapter includes the following sections:

Overview of Datatypes

A **datatype** is a classification of a particular type of information or data. Each value manipulated by Oracle has a datatype. The datatype of a value associates a fixed set of properties with the value. These properties cause Oracle to treat values of one datatype differently from values of another.

The datatypes recognized by Oracle are:

ANSI-supported datatypes

```
{ 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 datatypes

```
{ character_datatypes
| number_datatypes
| long_and_raw_datatypes
| datetime_datatypes
| large_object_datatypes
| rowid_datatypes
}
```

Oracle-supplied datatypes

```
{ any_types
| XML_types
| spatial_types
| media_types
| expression_filter_type
}
```

User-defined datatypes

User-defined datatypes use Oracle built-in datatypes and other user-defined datatypes to model the structure and behavior of data in applications

> **See Also:** Datatypes in *Oracle Database SQL Reference*

Oracle Built-In Datatypes

This section describes the kinds of Oracle built-in datatypes.

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 ]) ]
| BINARY_FLOAT
| BINARY_DOUBLE
}
```

rowid_datatypes

```
{ ROWID | UROWID [ (size) ] }
```

The codes listed for the datatypes are used internally by Oracle Database. The datatype code of a column or object attribute is returned by the `DUMP` function.

Table 6–1 Built-in Datatype Summary

Code	Datatype	Description
1	VARCHAR2(*size* [BYTE | CHAR])	Variable-length character string having maximum length *size* bytes or characters. Maximum *size* is 4000 bytes or characters, and minimum is 1 byte or 1 character. You must specify *size* for VARCHAR2. 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. 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 4000 bytes. You must specify *size* for NVARCHAR2.
2	NUMBER[(*precision* [, *scale*]])	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.
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 datatype contains the datetime fields YEAR, MONTH, DAY, HOUR, MINUTE, and SECOND. It does not have fractional seconds or a time zone.
21	BINARY_FLOAT	32-bit floating point number. This datatype requires 5 bytes, including the length byte.
22	BINARY_DOUBLE	64-bit floating point number. This datatype requires 9 bytes, including the length byte.
180	TIMESTAMP [(*fractional_seconds*)]	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_DATE_FORMAT parameter or implicitly by the NLS_TERRITORY parameter. The sizes varies from 7 to 11 bytes, depending on the precision. This datatype contains the datetime fields YEAR, MONTH, DAY, HOUR, MINUTE, and SECOND. It contains fractional seconds but does not have a time zone.
181	TIMESTAMP [(*fractional_seconds*)] 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_DATE_FORMAT parameter or implicitly by the NLS_TERRITORY parameter. The size is fixed at 13 bytes. This datatype contains the datetime fields YEAR, MONTH, DAY, HOUR, MINUTE, SECOND, TIMEZONE_HOUR, and TIMEZONE_MINUTE. It has fractional seconds and an explicit time zone.

Table 6–1 (Cont.) Built-in Datatype Summary

Code	Datatype	Description
231	TIMESTAMP [(*fractional_seconds*)] 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_DATE_FORMAT parameter or implicitly by the NLS_TERRITORY parameter. The sizes varies from 7 to 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*)]	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. Maximum *size* is 2000 bytes. You must specify *size* for a RAW value.
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 datatype 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. Maximum *size* is 2000 bytes or characters. Default and minimum *size* is 1 byte. BYTE and CHAR have the same semantics as for VARCHAR2.
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).

Table 6–1 (Cont.) Built-in Datatype Summary

Code	Datatype	Description
112	`NCLOB`	A character large object containing Unicode characters. Both fixed-width and variable-width character sets are supported, both using the database national character set. Maximum size is (4 gigabytes - 1) * (database block size). Stores national character set data.
113	`BLOB`	A binary large object. Maximum size is (4 gigabytes - 1) * (database block size).
114	`BFILE`	Contains a locator to a large binary file stored outside the database. Enables byte stream I/O access to external LOBs residing on the database server. Maximum size is 4 gigabytes.

See Also: Datatypes in *Oracle Database SQL Reference*

Oracle-Supplied Datatypes

This section describes the kinds of Oracle-supplied datatypes.

spatial_datatypes

```
{ SDO_Geometry | SDO_Topo_Geometry |SDO_GeoRaster }
```

Converting to Oracle Datatypes

SQL statements that create tables and clusters can also use ANSI datatypes and datatypes from the IBM products SQL/DS and DB2. Oracle recognizes the ANSI or IBM datatype name that differs from the Oracle datatype name, records it as the name of the datatype of the column, and then stores the column data in an Oracle datatype based on the conversions shown in the following table.

Table 6–2 ANSI Datatypes Converted to Oracle Datatypes

ANSI SQL Datatype	Oracle Datatype

`CHARACTER(n)`
`CHAR(n)`
	`CHAR(n)`

`CHARACTER VARYING(n)`
`CHAR VARYING(n)`
	`VARCHAR(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)` (**a**)	`NUMBER(p,s)`

Table 6–2 (Cont.) ANSI Datatypes Converted to Oracle Datatypes

ANSI SQL Datatype	Oracle Datatype
INTEGER INT SMALLINT	NUMBER(38)
FLOAT (**b**) DOUBLE PRECISION (**c**) REAL (**d**)	NUMBER

Notes:

- **a.** The NUMERIC and DECIMAL datatypes can specify only fixed-point numbers. For those datatypes, s defaults to 0.
- **b.** The FLOAT datatype is a floating-point number with a binary precision b. The default precision for this datatypes is 126 binary, or 38 decimal.
- **c.** The DOUBLE PRECISION datatype is a floating-point number with binary precision 126.
- **d.** The REAL datatype is a floating-point number with a binary precision of 63, or 18 decimal.

Table 6–3 SQL/DS and DB2 Datatypes Converted to Oracle Datatypes

SQL/DS or DB2 Datatype	Oracle Datatype
CHARACTER(n)	CHAR(n)
VARCHAR(n)	VARCHAR(n)
LONG VARCHAR(n)	LONG
DECIMAL(p,s) (**a**)	NUMBER(p,s)
INTEGER SMALLINT	NUMBER(38)
FLOAT (**b**)	NUMBER

Notes:

- **a.** The DECIMAL datatype can specify only fixed-point numbers. For this datatype, *s* defaults to 0..
- **b.** The FLOAT datatype is a floating-point number with a binary precision *b*. The default precision for this datatype is 126 binary or 38 decimal.

Do not define columns with the following SQL/DS and DB2 datatypes, because they have no corresponding Oracle datatype:

- GRAPHIC
- LONG VARGRAPHIC
- VARGRAPHIC
- TIME

Note that data of type TIME can also be expressed as Oracle datetime data.

See Also: Datatypes in *Oracle Database SQL Reference*

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:** Format Models in *Oracle Database SQL Reference*

Number Format Models

You can use number format models:

- In the `TO_CHAR` function to translate a value of `NUMBER` datatype to `VARCHAR2` datatype
- In the `TO_NUMBER` function to translate a value of `CHAR` or `VARCHAR2` datatype to `NUMBER` datatype

Number Format Elements

A number format model is composed of one or more number format elements. The following table lists the elements of a number format model.

Table 7–1 Number Format Elements

Element	Example	Description
, (comma)	`9,999`	Returns a comma in the specified position. You can specify multiple commas in a number format model. **Restrictions:** ■ A comma element cannot begin a number format model. ■ A comma cannot appear to the right of a decimal character or period in a number format model.
. (period)	`99.99`	Returns a decimal point, which is a period (.) in the specified position. **Restriction:** You can specify only one period in a number format model.
$	`$9999`	Returns value with a leading dollar sign.
0	`0999`	Returns leading zeros.
	`9990`	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`	Returns a value as Roman numerals in uppercase.
rn	`rn`	Returns a value as Roman numerals in lowercase. Value can be an integer between 1 and 3999.

Table 7–1 (Cont.) Number Format Elements

Element	Example	Description
S	`S9999`	Returns negative value with a leading minus sign (-). Returns positive value with a leading plus sign (+).
	`9999S`	Returns negative value with a trailing minus sign (-). Returns positive value with a trailing plus sign (+). **Restriction:** The S format element can appear only in the first or last position of a number format model.
TM	`TM`	The text minimum number format model returns (in decimal output) the smallest number of characters possible. This element is case insensitive. The default is TM9, which returns the number in fixed notation unless the output exceeds 64 characters. If the output exceeds 64 characters, then Oracle Database automatically returns the number in scientific notation. **Restrictions:** ■ You cannot precede this element with any other element. ■ You can follow this element only with one 9 or one E (or e), but not with any combination of these. The following statement returns an error: ■ `SELECT TO_CHAR(1234, 'TM9e') FROM DUAL;`
U	`U9999`	Returns in the specified position the Euro (or other) dual currency symbol (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 1 leading blank.

See Also: Number Format Models in *Oracle Database SQL Reference*

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	Specify in 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).
`DAY`	Yes	Name of day, padded with blanks to length of 9 characters.
`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 (the current value of the `NLS_DATE_FORMAT` parameter). Makes the appearance of the date components (day name, month number, and so forth) depend on the `NLS_TERRITORY` and `NLS_LANGUAGE` parameters. For example, in the `AMERICAN_AMERICA` locale, this is equivalent to specifying the format `'fmDay, Month dd, yyyy'`. In the `GERMAN_GERMANY` locale, it is equivalent to specifying the format '`fmDay, dd. Month yyyy`'. **Restriction:** You can specify this format only with the `TS` element, separated by white space.
`DS`	Yes	Returns a value in the short date format. Makes the appearance of the date components (day name, month number, and so forth) depend on the `NLS_TERRITORY` and `NLS_LANGUAGE` parameters. For example, in the `AMERICAN_AMERICA` locale, this is equivalent to specifying the format '`MM/DD/RRRR`'. In the `ENGLISH_UNITED_KINGDOM` locale, it is equivalent to specifying the format '`DD/MM/RRRR`'. **Restriction:** You can specify this format only with the `TS` element, separated by white space.
`DY`	Yes	Abbreviated name of day.
`E`	No	Abbreviated era name (Japanese Imperial, ROC Official, and Thai Buddha calendars).
`EE`	No	Full era name (Japanese Imperial, ROC Official, and Thai Buddha calendars).

Table 7–2 (Cont.) Datetime Format Elements

Element	Specify in TO_* datetime functions?	Description
`FF [1..9]`	Yes	Fractional seconds; no radix character is printed (use the X format element to add the radix character). Use the numbers 1 to 9 after FF to specify the number of digits in the fractional second portion of the datetime value returned. If you do not specify a digit, then Oracle Database uses the precision specified for the datetime datatype or the datatype's default precision. **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**: Additional discussion on this format model modifier in the *Oracle Database SQL Reference*
`FX`	Yes	Requires exact matching between the character data and the format model. **See Also**: Additional discussion on this format model modifier in the *Oracle Database SQL Reference*
`HH`	Yes	Hour of day (1-12).
`HH12`	No	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).
`MM`	Yes	Month (01-12; January = 01).
`MON`	Yes	Abbreviated name of month.
`MONTH`	Yes	Name of month, padded with blanks to length of 9 characters.
`PM` `P.M.`	No	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:** Additional discussion on `RR` datetime format element in the *Oracle Database SQL Reference*
`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.

Table 7–2 (Cont.) Datetime Format Elements

Element	Specify in TO_* datetime functions?	Description
TZD	Yes	Daylight savings information. The TZD value is an abbreviated time zone string with daylight savings information. It must correspond with the region specified in TZR. **Example:** PST (for US/Pacific standard time); PDT (for US/Pacific daylight time).
TZH	Yes	Time zone hour. (See TZM format element.) **Example:** 'HH:MI:SS.FFTZH:TZM'.
TZM	Yes	Time zone minute. (See TZH format element.) **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. **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.
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: Datetime Format Models in *Oracle Database SQL Reference*

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. In addition, it has a web-based user interface, *i*SQL*Plus.

The commands shown in Table A–1 are SQL*Plus commands available in the command-line interface. Not all commands or command parameters are shown.

See Also:

- *SQL*Plus Quick Reference*
- *SQL*Plus User's Guide and Reference*

Table A–1 Basic SQL*Plus Commands

Database Operation	SQL*Plus Command

Log in to SQL*Plus	`SQLPLUS [ { username[/passward][@connect_identifier] | / }`
	`[ AS { SYSDBA | SYSOPER } ]`
	`| /NOLOG`
	`]`

List help topics available in SQL*Plus	`HELP [ INDEX | topic ]`
Execute host commands	`HOST [ command ]`
Show SQL*Plus system variables or environment settings	`SHOW { ALL | ERRORS | USER | system_variable [, system_variable] ...}`

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

Database Operation	SQL*Plus Command
Alter SQL*Plus system variables or environment settings	SET *system_variable value*
Start up a database	STARTUP [PFILE = *filename*] [MOUNT [*dbname*] | NOMOUNT]

Connect to a database	CONNECT [{username[/password] [@connect_identifier] | /}
	      [AS {SYSOPER | SYSDBA}
	    |{proxy_user **[** username **]**
	      [/password] [@connect_identifier]}
	    ]
	
	Note: Brackets in boldface 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
Execute commands stored in the SQL buffer	/
List and execute commands stored in the SQL buffer	RUN

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

Database Operation	SQL*Plus Command
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 ]`

Log out of SQL*Plus	`{ EXIT | QUIT }`
	`  [ SUCCESS | FAILURE | WARNING ]`
	`  [ COMMIT | ROLLBACK ]`

Index