C

Oracle and Standard SQL

This appendix declares Oracle's conformance to the SQL standards established by the American National Standards Institute (ANSI) and the International Organization for Standardization (ISO).

The ISO SQL standard consists of eleven parts (SQL/Framework, SQL/Foundation, SQL/CLI, SQL/PSM, SQL/MED, SQL/OLB, SQL/Schemata, SQL/JRT, SQL/XML, SQL/MDA, and SQL/PGQ). The ANSI SQL standard consists of the same eleven parts.

The mandatory portion of SQL is known as Core SQL and is found in Part 2 (Foundation) and Part 11 (Schemata).

This appendix contains the following sections:

ANSI Standards

The following documents of the American National Standards Institute (ANSI) relate to SQL:

- INCITS/ANSI/ISO/IEC 9075-1:2016, Information technology—Database languages—SQL—Part 1: Framework (SQL/Framework)
- INCITS/ANSI/ISO/IEC 9075-2:2016, Information technology—Database languages—SQL—Part 2: Foundation (SQL/Foundation)
- INCITS/ANSI/ISO/IEC 9075-3:2016, Information technology—Database languages—SQL—Part 3: Call-Level Interface (SQL/CLI)

- INCITS/ANSI/ISO/IEC 9075-4:2016, Information technology—Database languages—SQL—Part 4: Persistent Stored Modules (SQL/PSM)
- INCITS/ANSI/ISO/IEC 9075-9:2016, Information technology—Database languages—SQL—Part 9: Management of External Data (SQL/MED)
- INCITS/ANSI/ISO/IEC 9075-10:2016, Information technology—Database languages—SQL—Part 10: Object Language Bindings (SQL/OLB)
- INCITS/ANSI/ISO/IEC 9075-11:2016, Information technology—Database languages—SQL—Part 11: Information and Definition Schemas (SQL/Schemata)
- INCITS/ANSI/ISO/IEC 9075-13:2016, Information technology—Database languages—SQL—Part 13: SQL Routines and Types using the Java Programming Language (SQL/JRT)
- INCITS/ANSI/ISO/IEC 9075-14:2016, Information technology—Database languages—SQL—Part 14: XML-Related Specifications (SQL/XML)
- INCITS/ANSI/ISO/IEC 9075-15:2023, Information technology—Database language SQL—Part 15: Multidimensional arrays (SQL/MDA)
- INCITS/ANSI/ISO/IEC 9075-16:2023, Information technology—Database language SQL—Part 16: Property Graph Queries (SQL/PGQ)

These standards are identical to the corresponding ISO standards listed in the next section.

You can obtain a copy of ANSI standards from this address:

American National Standards Institute
25 West 43rd Street, fourth floor
New York, NY 10036 USA
Telephone: +1.212.642.4900
Fax: +1.212.398.0023
Web site: http://www.ansi.org/

A subset of ANSI standards, including the SQL standard, are INCITS standards. You can obtain these from the InterNational Committee for Information Technology Standards (INCITS) at:

http://www.incits.org/

ISO Standards

The following documents of the International Organization for Standardization (ISO) relate to SQL:

- ISO/IEC 9075-1:2016, Information technology—Database languages—SQL—Part 1: Framework (SQL/Framework)
- ISO/IEC 9075-2:2016, Information technology—Database languages—SQL—Part 2: Foundation (SQL/Foundation)
- ISO/IEC 9075-3:2016, Information technology—Database languages—SQL—Part 3: Call-Level Interface (SQL/CLI)
- ISO/IEC 9075-4:2016, Information technology—Database languages—SQL—Part 4: Persistent Stored Modules (SQL/PSM)
- ISO/IEC 9075-9:2016, Information technology—Database languages—SQL—Part 9: Management of External Data (SQL/MED)
- ISO/IEC 9075-10:2016, Information technology—Database languages—SQL—Part 10: Object Language Bindings (SQL/OLB)

- ISO/IEC 9075-11:2016, Information technology—Database languages—SQL—Part 11: Information and Definition Schemas (SQL/Schemata)
- ISO/IEC 9075-13:2016, Information technology—Database languages—SQL—Part 13: SQL Routines and Types using the Java Programming Language (SQL/JRT)
- ISO/IEC 9075-14:2016, Information technology—Database languages—SQL—Part 14: XML-Related Specifications (SQL/XML)

You can obtain a copy of ISO standards from this address:

International Organization for Standardization
1, ch. de la Voie-Creuse
Case postale 56
CH-1211, Geneva 20, Switzerland
Phone: +41.22.749.0111
Fax: +41.22.733.3430
Web site: http://www.iso.org/

or from their Web store:

http://www.iso.org/iso/store.htm

Oracle Compliance to Core SQL

The ANSI and ISO SQL standards require conformance claims to state the type of conformance and the implemented facilities. The minimum claim of conformance is called Core SQL and is defined in Part 2, SQL/Foundation, and Part 11, SQL/Schemata, of the standard. The following products provide full or partial conformance with Core SQL as described in the tables that follow:

- Oracle Database server, release 12.2
- OTT (Oracle Type Translator), release 12.2
- Pro*C/C++, release 12.2
- Pro*COBOL, release 12.2

The SQL standards conformance features can be used either as a guide to portability, or as a guide to functionality. From the standpoint of portability, the user is interested in conformance to both the precise syntax and semantics of the standard feature. From the standpoint of functionality, the user is less concerned about the precise syntax and more concerned with issues of semantics. The tables in this appendix use the following terms regarding support for standard syntax and semantics:

- Full Support: The feature is supported with standard syntax and semantics.
- Partial Support: Some, but not all, of the standard syntax is supported; whatever is supported has standard semantics.
- Enhanced Support: The standard semantics is supported, as well as additional functionality.
- Equivalent Support: The standard semantics is supported using non-standard syntax.
- Similar Support: Neither the standard's syntax nor semantics are supported precisely, but similar functionality is provided.

Oracle's support for the features of Core SQL is listed in Table C-1:

Table C-1 Oracle Support of Core SQL Features

Feature ID	Feature Support
E011, Numeric data types	Oracle fully supports this feature.
E021, Character data types	Oracle fully supports these subfeatures: • E021-01, `CHARACTER` data type • E021-07, Character concatenation • E021-08, `UPPER` and `LOWER` functions • E021-09, `TRIM` function • E021-10, Implicit casting among character data types Oracle partially supports these subfeatures: • E021-02, `CHARACTER VARYING` data type (Oracle does not distinguish a zero-length `VARCHAR` string from `NULL`) • E021-03, Character literals (Oracle regards the zero-length literal '' as being null) • E021-12, Character comparison (Oracle's rules for padding the shorter of two strings to be compared differs from the standard) Oracle has equivalent functionality for these subfeatures: • E021-04, `CHARACTER_LENGTH` function: use `LENGTH` function instead • E021-05, `OCTET_LENGTH` function: use `LENGTHB` function instead • E021-06, `SUBSTRING` function: use `SUBSTR` function instead • E021-11, `POSITION` function: use `INSTR` function instead
E031, Identifiers	Oracle supports this feature, with the following exceptions: • Oracle does not support the escape sequence to permit a double quote within a quoted identifier • A non-quoted identifier may not be equivalent to an Oracle reserved word (the list of Oracle reserved words differs from the standard's list) • A column name may not be `ROWID`, even as a quoted identifier Oracle extends this feature as follows: • An identifier may be up to 128 characters long • A non-quoted identifier may have dollar sign ($) or pound sign (#)
E051, Basic query specification	Oracle fully supports the following subfeatures: • E051-01, `SELECT DISTINCT` • E051-02, `GROUP BY` clause • E051-04, `GROUP BY` can contain columns not in `SELECT` list • E051-05, `SELECT` list items can be renamed • E051-06, `HAVING` clause • E051-07, Qualified * in `SELECT` list Oracle partially supports the following subfeatures: • E051-08, Correlation names in `FROM` clause (Oracle supports correlation names, but not the optional `AS` keyword) Oracle has equivalent functionality for the following subfeature: • E051-09, Rename columns in the `FROM` clause (column names can be renamed in a subquery in the `FROM` clause)
E061, Basic predicates and search conditions	Oracle fully supports this feature, except that Oracle comparison of character strings differs from the standard as follows: In the standard, two character strings of unequal length are compared by either padding the shorter string with spaces or a fictitious character that is less than all actual characters. The decision on padding is made on the basis of the character set. In Oracle, the decision is based on whether the comparands are of fixed or varying length.

Table C-1 (Cont.) Oracle Support of Core SQL Features

Feature ID	Feature Support
E071, Basic query expressions	Oracle fully supports the following subfeatures: • E071-01, `UNION DISTINCT` table operator • E071-02, `UNION ALL` table operator • E071-05, Columns combined by table operators need not have exactly the same type • E071-06, table operators in subqueries Oracle has equivalent functionality for the following subfeature: • E071-03, `EXCEPT DISTINCT` table operator: Use `MINUS` instead of `EXCEPT DISTINCT`
E081, Basic privileges	Oracle fully supports all subfeatures of this feature, except E081-09, `USAGE` privileges. In the standard, the `USAGE` privilege permits the user to use domains, collations, character sets, transliterations, user-defined types and sequence generators. Oracle does not support domains or transliterations. No privileges are required to access collations and character sets. The Oracle privilege to use a user-defined type is `EXECUTE`. The Oracle privilege to use a sequence type is `SELECT`.
E091, Set functions	Oracle fully supports this feature.
E101, Basic data manipulation	Oracle fully supports this feature.
E111, Single row `SELECT` statement	Oracle fully supports this feature.
E121, Basic cursor support	Oracle fully supports the following subfeatures: • E121-02, `ORDER BY` columns need not be in `SELECT` list • E121-03, Value expressions in `ORDER BY` clause • E121-04, `OPEN` statement • E121-06, Positioned `UPDATE` statement • E121-07, Positioned `DELETE` statement • E121-08, `CLOSE` statement Oracle provides partial support for the following subfeatures: • E121-01, `DECLARE CURSOR` - fully supported, except for the `FOR READ ONLY` syntax • E121-10 `FETCH` statement, implicit `NEXT` - fully supported, except for the noise word `FROM` Oracle provides enhanced support for the following subfeature: • E121-17, `WITH HOLD` cursors (in the standard, a cursor is not held through a `ROLLBACK`, but Oracle does hold through `ROLLBACK`)
E131, Null value support	Oracle fully supports this feature, with this exception: In Oracle, a null of character type is indistinguishable from a zero-length character string.
E141, Basic integrity constraints	Oracle fully supports this feature.
E151, Transaction support	Oracle fully supports this feature.
E152, Basic `SET TRANSACTION` statement	Oracle fully supports this feature.

Table C-1 (Cont.) Oracle Support of Core SQL Features

Feature ID	Feature Support
E153, Updatable queries with subqueries	Oracle fully supports this feature.
E161, SQL comments using leading double minus	Oracle fully supports this feature.
E171, SQLSTATE support	Oracle fully supports this feature.
E182, Host language binding	Oracle fully supports this feature through Pro*C/C++ and Pro*COBOL
F021, Basic information schema	Oracle does not have any of the views in this feature. However, Oracle makes the same information available in other metadata views: • Instead of `TABLES`, use `ALL_TABLES`. • Instead of `COLUMNS`, use `ALL_TAB_COLUMNS`. • Instead of `VIEWS`, use `ALL_VIEWS`. However, Oracle's `ALL_VIEWS` does not display whether a user view was defined `WITH CHECK OPTION` or if it is updatable. To see whether a view has `WITH CHECK OPTION`, use `ALL_CONSTRAINTS`, with `TABLE_NAME` equal to the view name and look for `CONSTRAINT_TYPE` equal to 'V'. • Instead of `TABLE_CONSTRAINTS`, `REFERENTIAL_CONSTRAINTS`, and `CHECK_CONSTRAINTS`, use `ALL_CONSTRAINTS`. However, Oracle's `ALL_CONSTRAINTS` does not display whether a constraint is deferrable or initially deferred.
F031, Basic schema manipulation	Oracle fully supports these subfeatures: • F031-01, `CREATE TABLE` statement to create persistent base tables • F031-02, `CREATE VIEW` statement • F031-03, `GRANT` statement Oracle provides equivalent support for this subfeature: • F031-04, `ALTER TABLE` statement: `ADD COLUMN` clause (Oracle does not support the optional keyword `COLUMN` in this syntax. Also, Oracle requires the column definition to be enclosed in parentheses, unlike the standard.) Oracle does not support these subfeatures (because Oracle does not support the keyword `RESTRICT`): • F031-13, `DROP TABLE` statement: `RESTRICT` clause • F031-16, `DROP VIEW` statement: `RESTRICT` clause • F031-19, `REVOKE` statement: `RESTRICT` clause (Oracle `DROP` commands enhance the standard by invalidating dependent objects, so that they can be subsequently revalidated without user action, rather than either cascading all drops to dependent objects or prohibiting a drop if there is a dependent object.)
F041, Basic joined table	Oracle fully supports this feature.
F051, Basic date and time	Oracle fully supports this feature, except the following subfeatures are not supported: • F051-02, `TIME` data type • F051-07, `LOCALTIME`

Table C-1 (Cont.) Oracle Support of Core SQL Features

Feature ID	Feature Support
F081, `UNION` and `EXCEPT` in views	Oracle fully supports `UNION` in views.
F131, Grouped operations	Oracle fully supports this feature.
F181, Multiple module support	Oracle fully supports this feature.
F201, `CAST` function	Oracle fully supports this feature.
F221, Explicit defaults	Oracle's `DEFAULT ON NULL` capability in a column definition provides equivalent functionality for the `INSERT` statement though not for the `UPDATE` statement.
F261, `CASE` expressions	Oracle fully supports this feature.
F311, Schema definition statement	Oracle fully supports this feature.
F471, Scalar subquery values	Oracle fully supports this feature.
F481, Expanded null predicate	Oracle fully supports this feature.
F501, Feature and conformance views	Oracle does not support this feature.
F812, Basic flagging	Oracle has a flagger, but it flags SQL-92 compliance rather than SQL:2011 compliance.
S011, Distinct types	Distinct types are strongly typed scalar types. A distinct type can be emulated in Oracle using an object type with only one attribute. The standard's Information Schema view called `USER_DEFINED_TYPES` is equivalent to Oracle's metadata view `ALL_TYPES`.

Table C-1 (Cont.) Oracle Support of Core SQL Features

Feature ID	Feature Support
T321, Basic SQL-invoked routines	Oracle fully supports these subfeatures: • T321-03, function invocation • T321-04, `CALL` statement Oracle supports these subfeatures with syntactic differences: • T321-01, user-defined functions with no overloading • T321-02, user-defined procedures with no overloading The Oracle syntax for `CREATE FUNCTION` and `CREATE PROCEDURE` differs from the standard as follows: • In the standard, the mode of a parameter (`IN`, `OUT`, or `INOUT`) comes before the parameter name, whereas in Oracle it comes after the parameter name. • The standard uses `INOUT`, whereas Oracle uses `IN OUT`. • Oracle requires either `IS` or `AS` after the return type and before the definition of the routine body, while the standard lacks these keywords. • If the routine body is in C (for example), then the standard uses the keywords `LANGUAGE C EXTERNAL NAME` to name the routine, whereas Oracle uses `LANGUAGE C NAME`. • If the routine body is in SQL, then Oracle uses its proprietary procedural extension called PL/SQL. Oracle supports the following subfeature in PL/SQL but not in Oracle SQL: • T321-05, `RETURN` statement Oracle provides equivalent functionality for the following subfeatures: • T321-06, `ROUTINES` view: Use the `ALL PROCEDURES` metadata view. • T321-07, `PARAMETERS` view: Use the `ALL_ARGUMENTS` and `ALL_METHOD_PARAMS` metadata views.
T631, `IN` predicate with one list element	Oracle fully supports this feature.

Oracle Support for Optional Features of SQL/Foundation

Oracle's support for optional features of SQL/Foundation is listed in Table C-2:

Table C-2 Oracle Support for Optional Features of SQL/Foundation

Feature ID	Feature Support
B012, Embedded C	Oracle fully supports this feature.
B013, Embedded COBOL	Oracle fully supports this feature.
B021, Direct SQL	Oracle fully supports this feature, as SQL*Plus.

Table C-2 (Cont.) Oracle Support for Optional Features of SQL/Foundation

Feature ID	Feature Support
B031, Basic dynamic SQL	Oracle supports dynamic SQL in two styles, documented in the embedded language manuals as "Oracle dynamic SQL" and "ANSI dynamic SQL." ANSI dynamic SQL is an implementation of the standard, with the following restrictions: • Oracle supports a subset of the descriptor items. • For <input using clause>, Oracle only supports <using input descriptor>. • For <output using clause>, Oracle only supports <into descriptor>. • Dynamic parameters are indicated by a colon followed by an identifier rather than a question mark. Oracle dynamic SQL is similar to standard dynamic SQL, with the following modifications: • Parameters are indicated by a colon followed by an identifier, instead of a question mark. • Oracle's `DESCRIBE SELECT LIST FOR` statement replaces the standard's `DESCRIBE OUTPUT`. • Oracle provides `DECLARE STATEMENT` if you want to declare a cursor using a dynamic SQL statement physically prior to the `PREPARE` statement that prepares the dynamic SQL statement.
B032, Extended dynamic SQL	In ANSI dynamic SQL, Oracle only implements the ability to declare global statements and global cursors from this feature; the rest of the feature is not supported. In Oracle dynamic SQL, Oracle's `DESCRIBE BIND VARIABLES` is equivalent to the standard's `DESCRIBE INPUT`; the rest of this feature is not supported.
B122, Routine language C	Oracle supports external routines written in C, though Oracle does not support the standard syntax for creating such routines.
B128, Routine language SQL	Oracle supports routines written in PL/SQL, which is Oracle's equivalent to the standard procedural language SQL/PSM.
F032, `CASCADE` drop behavior	In Oracle, a `DROP` command invalidates all of the dropped object's dependent objects. Invalidated objects are effectively unusable until the dropped object is redefined in such a way to allow successful recompilation of the invalidated object.
F033, `ALTER TABLE` statement: `DROP COLUMN` clause	Oracle provides a `DROP COLUMN` clause, but without the `RESTRICT` or `CASCADE` options found in the standard.
F034, Extended `REVOKE` statement	Oracle supports the following parts of this feature: • F034-01, `REVOKE` statement performed by other than the owner of a schema object • F034-03, `REVOKE` statement to revoke a privilege that the grantee has `WITH GRANT OPTION` Oracle provides equivalent functionality for the following parts of this feature: • `CASCADE`: In Oracle, a `REVOKE` invalidates all dependent objects, which become effectively unusable until the metadata is changed through subsequent `CREATE` and `GRANT` commands enabling the invalidated object to be successfully recompiled.
F052, Intervals and datetime arithmetic	Oracle only supports the `INTERVAL YEAR TO MONTH` and `INTERVAL DAY TO SECOND` data types.

Table C-2 (Cont.) Oracle Support for Optional Features of SQL/Foundation

Feature ID	Feature Support
F111, Isolations levels other than SERIALIZABLE	In addition to SERIALIZABLE, Oracle supports the READ COMMITTED isolation level.
F121, Basic diagnostics management	Much of the functionality of this feature is provided through the SQLCA in embedded languages.
F191, Referential delete actions	Oracle supports ON DELETE CASCADE and ON DELETE SET NULL.
F200, TRUNCATE TABLE	Oracle fully supports this feature, and extends it by permitting truncation of a table that references itself in a referential integrity constraint, and the ability to cascade to child tables with enabled ON DELETE CASCADE referential constraints.
F231, Privilege tables	Oracle makes this information available in the following metadata views: • Instead of TABLE_PRIVILEGES, use ALL_TAB_PRIVS. • Instead of COLUMN_PRIVILEGES, use ALL_COL_PRIVS. • Oracle does not support USAGE privileges so there is no equivalent to USAGE_PRIVILEGES.
F281, LIKE enhancements	Oracle fully supports this feature.

F291, UNIQUE predicate	The IS A SET condition may be used to test whether a multiset is a set; that is, each row is unique. Thus, the equivalent of
	`UNIQUE <table subquery>`
	is
	`CAST (<table subquery> AS MULTISET) IS A SET`

F302, INTERSECT table operator	Syntactically, Oracle differs from the standard in that UNION, INTERSECT, and MINUS have the same precedence.
F312, MERGE statement	The Oracle MERGE statement is almost the same as the standard, with these exceptions: • Oracle does not support the optional AS keyword before a table alias. • Oracle does not support the ability to rename columns of the table specified in the USING clause with a parenthesized list of column names following the table alias. • Oracle does not support the <override clause>.
F314, MERGE statement with DELETE branch	Oracle has similar functionality, though in Oracle you must first update a row, after which you can delete it if the revised row meets a condition.
F321, User authorization	Oracle provides equivalent functionality for the following subfeatures: • Use SYS_CONTEXT ('USERENV', 'SESSION_USER') instead of SESSION_USER • Use SYS_CONTEXT ('USERENV', 'CURRENT_USER') instead of CURRENT_USER Oracle does not support the following subfeatures: • SYSTEM_USER • SET SESSION AUTHORIZATION statement

Table C-2 (Cont.) Oracle Support for Optional Features of SQL/Foundation

Feature ID	Feature Support
F341, Usage tables	Oracle makes this information available in the views `ALL_DEPENDENCIES`, `DBA_DEPENDENCIES`, and `USER_DEPENDENCIES`.
F381, Extended schema manipulation	Oracle fully supports the following element of this feature: • Oracle supports the standard syntax to add a table constraint using `ALTER TABLE`. Oracle partially supports the following element of this feature: • Oracle supports the standard syntax to drop a table constraint, except that Oracle does not support `RESTRICT`. Oracle provides equivalent functionality for the following element of this feature: • To alter the default value of a column, use the `MODIFY` option of `ALTER TABLE`. Oracle does not support the following parts of this feature: • `DROP SCHEMA` statement • `ALTER ROUTINE` statement
F382, Alter column data type	Oracle supports this functionality, though with non-standard syntax. As an extension to the standard, Oracle allows you to reduce the size or precision of a column.
F383, Set column not null clause	Oracle provides equivalent functionality for the two subfeatures of this feature: • To add a `NOT NULL` constraint to an existing column, use `ALTER TABLE ... MODIFY` • To drop a `NOT NULL` constraint, use `ALTER TABLE` to drop the constraint by name
F384, Drop identity property clause	Oracle provides equivalent functionality using `ALTER TABLE ... MODIFY (... DROP IDENTITY)`
F386, Set identity column generation clause	Oracle provides equivalent functionality. Oracle's syntax and semantics are the same as the standard, with this exception: • Oracle does not support `RESTART`; use `START WITH` instead. When restarting an identity column, the values of the other parameters for the identity column are reset to their defaults unless explicitly set in the `ALTER TABLE` statement. Oracle's `START WITH LIMIT VALUE` option is an extension on the standard.
F391, Long identifiers	Oracle supports identifiers up to 128 characters in length.
F393, Unicode escapes in literals	The Oracle `UNISTR` function supports numeric escape sequences for all Unicode characters.

Table C-2 (Cont.) Oracle Support for Optional Features of SQL/Foundation

Feature ID	Feature Support
F394, Optional normal form specification	This feature adds the keywords NFC, NFD, NFKC, and NKD to the NORMALIZE function and the IS NORMAL predicate. Without these keywords, NFC is the default (see Feature T061, UCS support). Oracle supports all four normalization forms, with nonstandard syntax, as follows: • For NFC, use COMPOSE • For NFD, use DECOMPOSE with the CANONICAL option • For NFKD, use DECOMPOSE with the COMPATIBILITY option • For NFKC, use DECOMPOSE with the CANONICAL option followed by COMPOSE Oracle does not support the IS NORMAL predicate.
F401, Extended joined table	Oracle supports FULL outer joins, CROSS joins, and NATURAL joins.
F402, Named column joins for LOBs, arrays and multisets	Oracle supports named column joins for columns whose declared type is nested table. Oracle does not support named column joins for LOBs or arrays.
F403, Partitioned join tables	Oracle supports this feature, except with FULL outer joins.
F411, Time zone specification	Oracle fully supports TIMESTAMP WITH TIME ZONE, but does not support TIME WITH TIME ZONE.
F421, National character	Oracle fully supports this feature.
F431, Read-only scrollable cursors	Oracle fully supports this feature.
F441, Extended set function support	Oracle supports the following parts of this feature: • The ability in the WHERE clause to reference a column that is defined using an aggregate, either in a view or an inline view • COUNT without DISTINCT of an expression • Aggregates that reference columns that are outer references with respect to the aggregating query. However, Oracle defines the aggregating query as the innermost query containing the aggregate, rather than the innermost query that defines a range variable referenced in the aggregate.
F442, Mixed column references in set functions	Oracle fully supports this feature.
F461, Named character sets	Oracle supports many character sets with Oracle-defined names. Oracle does not support any other aspect of this feature.
F491, Constraint management	Oracle fully supports this feature.
F492, Optional table constraint enforcement	ENFORCED in the standard is equivalent to ENABLE VALIDATE in Oracle. NOT ENFORCED in the standard is equivalent to DISABLE NOVALIDATE in Oracle. Other combinations of the ENABLE | DISABLE, VALIDATE | NOVALIDATE, and RELY | NORELY options are extensions of the standard.
F531, Temporary tables	Oracle supports GLOBAL TEMPORARY tables.
F555, Enhanced seconds precision	Oracle provides enhanced support for this feature, supporting up to 9 places after the decimal point.
F561, Full value expressions	Oracle fully supports this feature.
F571, Truth value tests	Oracle's LNNVL function is equivalent to the standard's IS NOT TRUE predicate.

Table C-2 (Cont.) Oracle Support for Optional Features of SQL/Foundation

Feature ID	Feature Support
F591, Derived tables	Oracle supports <derived table>, with the exception of: • Oracle does not support the optional AS keyword before a table alias. • Oracle does not support <derived column list>.
F641, Row and table constructors	In Oracle, a row constructor may be used in an equality or inequality comparison with another row constructor or with a subquery. Oracle does not support anything else in this feature.
F690, Collation support	Oracle's NLSSORT function may be used to change the collation of character expressions.
F693, SQL-sessions and client module collations	To set a session collation, use ALTER SESSION SET NLS_COMP = 'LINGUISTIC' and also set NLS_SORT to your desired collation. Oracle does not support client module collations.
F695, Translation support	The Oracle CONVERT function can convert between the database character set and the national character set. For other character sets, store the data in the RAW data type and use the PL/SQL package function UTL_RAW.CONVERT. Oracle does not provide the ability to add or drop character set conversions.
F721, Deferrable constraints	Oracle fully supports this feature.
F731, INSERT column privileges	Oracle fully supports this feature.
F761, Session management	Oracle provides the following equivalents for elements of this feature: • The equivalent to the standard's SET SESSION CHARACTERISTICS AS TRANSACTION SERIALIZABLE is ALTER SESSION SET ISOLATION_LEVEL = SERIALIZABLE. • The equivalent to the standard's SET SCHEMA is ALTER SESSION SET CURRENT_SCHEMA. • The equivalent to the standard's SET COLLATION is ALTER SESSION SET NLS_SORT.
F763, CURRENT_SCHEMA	Oracle's equivalent is SYS_CONTEXT ('USERENV', 'CURRENT_SCHEMA')
F771, Connection management	Oracle's CONNECT statement provides the same functionality as the standard's CONNECT statement, though with different syntax. Instead of using the standard's SET CONNECTION, Oracle provides the AT clause to indicate which connection a SQL statement should be performed on. Oracle embedded languages let you disconnect from a connection by using the RELEASE option of either COMMIT or ROLLBACK.
F781, Self-referencing operations	Oracle fully supports this feature.
F801, Full set function	Oracle fully supports this feature.
F831, Full cursor update	Oracle supports the combination of FOR UPDATE and ORDER BY clauses in a query.
F841, LIKE_REGEX predicate	Oracle's equivalent is REGEXP_LIKE. Oracle's pattern syntax lacks some of the features of the standard's. Oracle's match parameter has the same capabilities as the standard's, though with a few spelling differences.

Table C-2 (Cont.) Oracle Support for Optional Features of SQL/Foundation

Feature ID	Feature Support
F842, `OCCURRENCES_REGEX` function	Oracle's equivalent is `REGEXP_COUNT`. Oracle's pattern syntax lacks some of the features of the standard's. Oracle's match parameter has the same capabilities as the standard's, though with a few spelling differences.
F843, `POSITION_REGEX` function	Oracle's equivalent is `REGEXP_INSTR`. Oracle's pattern syntax lacks some of the features of the standard's. Oracle's match parameter has the same capabilities as the standard's, though with a few spelling differences.
F844, `SUBSTRING_REGEX` function	Oracle's equivalent is `REGEXP_SUBSTR`. Oracle's pattern syntax lacks some of the features of the standard's. Oracle's match parameter has the same capabilities as the standard's, though with a few spelling differences.
F845, `TRANSLATE_REGEX` function	Oracle's equivalent is `REGEXP_REPLACE`. Oracle's pattern syntax lacks some of the features of the standard's. Oracle's match parameter has the same capabilities as the standard's, though with a few spelling differences.
F850, Top-level <order by clause> in <query expression>	Oracle fully supports this feature.
F851, <order by clause> in subqueries	Oracle fully supports this feature.
F852, Top-level <order by clause> in views	Oracle fully supports this feature.
F855, Nested <order by clause> in <query expression>	Oracle fully supports this feature.
F856, Nested <fetch first clause> in <query expression>	Oracle fully supports this feature.
F857, Top-level <fetch first clause> in a <query expression>	Oracle fully supports this feature.
F858, <fetch first clause> in subqueries	Oracle fully supports this feature.
F859, Top-level <fetch first clause> in views	Oracle fully supports this feature.
F860, Dynamic <fetch first row count> in <fetch first clause>	Oracle fully supports this feature.
F861, Top-level <result offset clause> in <query expression>	Oracle fully supports this feature.
F862, <result offset clause> in subqueries	Oracle fully supports this feature.
F863, Nested <result offset clause> in <query expression>	Oracle fully supports this feature.
F864, Top-level <result offset clause> in views	Oracle fully supports this feature.
F865, Dynamic <offset row count> in <result offset clause>	Oracle fully supports this feature.
F866, `FETCH FIRST` clause: `PERCENT` option	Oracle fully supports this feature.

Table C-2 (Cont.) Oracle Support for Optional Features of SQL/Foundation

Feature ID	Feature Support
F867, `FETCH FIRST` clause: `WITH TIES` option	Oracle fully supports this feature.
R010, Row pattern recognition: `FROM` clause	Oracle fully supports this feature.
S023, Basic structured types	Oracle's object types are equivalent to structured types in the standard.
S024, Enhanced structured types	Oracle's syntax is non-standard, but provides equivalents for the following: • `NOT INSTANTIABLE` • `STATIC` methods • `RELATIVE`, `MAP`, and `STATE` orderings. The keyword in Oracle for `RELATIVE` orderings is `ORDER`. There is no keyword for `STATE` orderings (this is the default, if no other ordering is defined). Unlike the standard, Oracle does not support `EQUALS ONLY` on non-`STATE` orderings. (See also Feature S251, User-defined orderings.) • `SELF AS RESULT` in the signature of constructor methods
S025, Final structured types	Oracle's final object types are equivalent to final structured types in the standard.
S026, Self-referencing structured types	In Oracle, an object type OT may have a reference that references OT.
S041, Basic reference types	Oracle's reference types are equivalent to reference types in the standard. To dereference a reference, dot notation is used, instead of -> as in the standard.
S043, Enhanced reference types	Oracle supports the following elements of this feature: • `DEREF` operator to return the object referenced by a reference • `SCOPE` clause as a constraint on columns of tables or materialized views • Adding and dropping the scope of a column • References that are either system-generated or derived from the primary key (but not from any other list of columns, nor from a list of attributes of the type)
S051, Create table of type	Oracle's object tables are equivalent to tables of structured type in the standard.
S081, Subtables	Oracle supports hierarchies of object views, but not of object base tables. To emulate a hierarchy of base tables, create a hierarchy of views on those base tables.
S091, Basic array support	Oracle `VARRAY` types are equivalent to array types in the standard. However, Oracle does not support storage of arrays of LOBs. To access a single element of an array using a subscript, you must use PL/SQL. Oracle supports the following aspects of this feature with nonstandard syntax: • To construct an instance of varray type, including an empty array, use the varray type constructor. • To unnest a varray in the `FROM` clause, use the `TABLE` operator. • To get the cardinality of a varray, use the `COUNT` method in PL/SQL.

Table C-2 (Cont.) Oracle Support for Optional Features of SQL/Foundation

Feature ID	Feature Support
S092, Arrays of user-defined types	Oracle supports `VARRAY`s of object types.
S094, Arrays of reference types	Oracle supports `VARRAY`s of references.
S095, Array constructors by query	Oracle supports this using `CAST` (`MULTISET` (`SELECT` ...) `AS` *`varray_type`*). The ability to order the elements of the array using `ORDER BY` is not supported.
S097, Array element assignment	In PL/SQL, you can assign to array elements, using syntax that is similar to the standard (SQL/PSM).
S098, `ARRAY_AGG`	Oracle does not have an aggregate that results in a varray. Instead, the `COLLECT` aggregate may be used to create a multiset, which can be cast to an array of the element type.
S111, `ONLY` in query expressions	Oracle supports the `ONLY` clause for view hierarchies; Oracle does not support hierarchies of base tables.
S151, Type predicate	Oracle fully supports this feature.
S161, Subtype treatment	Oracle fully supports this feature.
S162, Subtype treatment for references	Supported, with a minor syntactic difference: The standard requires parentheses around the referenced type's name; Oracle does not support parentheses in this position.
S201, SQL-invoked routines on arrays	PL/SQL provides the ability to pass arrays as parameters and return arrays as the result of functions. Procedures and functions written in C may pass arrays and return arrays as the result of functions using the Oracle Type Translator (OTT).
S202, SQL-invoked routines on multisets	A PL/SQL routine may have nested tables as parameters, and may return a nested table. Routines written in C may pass arrays and return arrays as the result of functions using the Oracle Type Translator.
S232, Array locators	Oracle Type Translator supports descriptors for arrays, which achieve the same purpose as locators.
S233, Multiset locators	Oracle supports locators for nested tables.
S241, Transform functions	The Oracle Type Translator provides the same capability as transforms.
S251, User-defined orderings	Oracle's object type ordering capabilities correspond to the standard's capabilities as follows: • Oracle's `MAP` ordering corresponds to the standard's `ORDER FULL BY MAP` ordering. • Oracle's `ORDER` ordering corresponds to the standard's `ORDER FULL BY RELATIVE` ordering. • If an Oracle object type has neither `MAP` nor `ORDER` declared, then this corresponds to `EQUALS ONLY BY STATE` in the standard. • Oracle does not have unordered object types; you can alter the ordering but you cannot drop it.
S261, Specified type method	The `GetTypeName` method of the `ANYDATA` type may be used to learn the name of a type.

Table C-2 (Cont.) Oracle Support for Optional Features of SQL/Foundation

Feature ID	Feature Support
S271, Basic multiset support	Multisets in the standard are supported as nested table types in Oracle. The Oracle nested table data type based on a scalar type ST is equivalent, in standard terminology, to a multiset of rows having a single field of type ST and named *column_value*. The Oracle nested table type based on an object type is equivalent to a multiset of structured type in the standard. Oracle supports the following elements of this feature on nested tables using the same syntax as the standard has for multisets: • The CARDINALITY function • The SET function • The MEMBER predicate • The IS A SET predicate • The COLLECT aggregate All other aspects of this feature are supported with non-standard syntax, as follows: • To create an empty multiset, denoted MULTISET[] in the standard, use an empty constructor of the nested table type. • To obtain the sole element of a multiset with one element, denoted ELEMENT (<multiset value expression>) in the standard, use a scalar subquery to select the single element from the nested table. • To construct a multiset by enumeration, use the constructor of the nested table type. • To construct a multiset by query, use CAST with a multiset argument, casting to the nested table type. • To unnest a multiset, use the TABLE operator in the FROM clause.
S272, Multisets of user-defined types	Oracle's nested table type permits a multiset of structured types. Oracle does not have distinct types, so a multiset of distinct types is not supported.
S274, Multisets of reference types	A nested table type can have one or more columns of reference type.
S275, Advanced multiset support	Oracle supports the following elements of this feature on nested tables using the same syntax as the standard has for multisets: • The MULTISET UNION, MULTISET INTERSECTION, and MULTISET EXCEPT operators • The SUBMULTISET predicate • = and <> predicates Oracle does not support the FUSION or INTERSECTION aggregates.
S281, Nested collection types	Oracle permits nesting of its collection types (varray and nested table).
S401, Distinct types based on array types	Oracle's varray types are strongly typed.
S403, ARRAY_MAX_CARDINALITY	In PL/SQL, the LIMIT method of a varray returns its maximum cardinality.
S404, TRIM_ARRAY	In PL/SQL, the TRIM method of a varray can be used to trim the varray.

Table C-2 (Cont.) Oracle Support for Optional Features of SQL/Foundation

Feature ID	Feature Support
T041, Basic LOB data type support	Oracle supports the following aspects of this feature: • The keywords BLOB, CLOB, and NCLOB • Concatenation, UPPER, LOWER on CLOBs Oracle provides equivalent support for the following aspects of this feature: • Use INSTR instead of POSITION. • Use LENGTH instead of CHAR_LENGTH. Oracle does not support the following aspects of this feature: • The keywords BINARY LARGE OBJECT, CHARACTER LARGE OBJECT, and NATIONAL CHARACTER LARGE OBJECT as synonyms for BLOB, CLOB, and NCLOB, respectively • <binary string literal> • The ability to specify an upper bound on the length of a BLOB or CLOB • Concatenation of BLOBs
T042, Extended LOB support	Oracle fully supports the following element of this feature: • TRIM function on a CLOB argument Oracle provides equivalent functionality for the following elements of this feature: • BLOB and CLOB substring, supported using SUBSTR • SIMILAR predicate, supported using REGEXPR_LIKE to perform pattern matching with a Perl-like syntax The following elements of this feature are not supported: • Comparison predicates with BLOB or CLOB operands • CAST with a BLOB or CLOB operand • OVERLAY (This may be emulated using SUBSTR and string concatenation.) • LIKE predicate with BLOB or CLOB operands
T051, Row types	Oracle object types can be used in place of the standard's row types.
T061, UCS support	Oracle provides equivalent functionality for the following elements of this feature: • Oracle supports the keyword CHAR instead of CHARACTERS, and BYTE instead of OCTETS, in a character data type declaration. • The Oracle COMPOSE function is equivalent to the standard's NORMALIZE function. Oracle does not support the IS NORMALIZED predicate.
T071, BIGINT data type	On many implementations, BIGINT refers to a binary integer type with 64 bits, which supports almost 19 decimal digits. The Oracle NUMBER type supports 39 decimal digits.
T111, Updatable joins, unions and columns	Oracle's updatable join views are similar to the standard's updatable join capabilities. Unlike the standard, Oracle does not require an updatable join view to display the strong candidate key in the SELECT list. Although an updatable join view might have more than one key-preserved table, only one of them may be modified using an UPDATE or DELETE, unlike the standard, which modifies all key-preserved tables of an updatable join.

Table C-2 (Cont.) Oracle Support for Optional Features of SQL/Foundation

Feature ID	Feature Support
T121, WITH (excluding RECURSIVE) in query expression	Oracle fully supports this feature.
T122, WITH (excluding RECURSIVE) in subquery	Oracle fully supports this feature.
T131, Recursive query	Oracle supports the use of a WITH clause element that references itself, but without the RECURSIVE keyword. Alternatively, Oracle's START WITH and CONNECT BY clauses can be used to perform many recursive queries.
T132, Recursive query in subquery	Oracle supports the use of a WITH clause element that references itself, but without the RECURSIVE keyword. Alternatively, Oracle's START WITH and CONNECT BY clauses can be used to perform many recursive queries.
T141, SIMILAR predicate	Oracle provides REGEXP_LIKE for pattern matching with a Perl-like syntax.
T172, AS subquery clause in table definition	Oracle's AS subquery feature of CREATE TABLE has substantially the same functionality as the standard, though there are some syntactic differences.
T174, Identity columns	Oracle supports this feature, with the following syntactic differences: • Oracle uses NOMINVALUE and NOMAXVALUE instead of the standard's NO MINVALUE and NO MAXVALUE. • To restart an identity column, in an ALTER TABLE MODIFY statement, use START WITH LIMIT VALUE to restart at the highest value (for an increasing identity column) or the lowest value (for a decreasing identity column); use START WITH number to restart at a specific number. GENERATED BY DEFAULT ON NULL is an Oracle extension.
T175, Generated columns	Oracle supports this feature, with the following restrictions: • Generated columns are not supported in temporary tables. • The data type of a generated column may not be LOB or XML.
T176, Sequence generator support	Oracle's sequences have the same capabilities as the standard's, though with different syntax.
T178, Identity columns: simple restart option	Oracle's START WITH LIMIT VALUE is the same as the standard's simple restart if the identity column has not cycled.
T180, System-versioned tables	Oracle's Flashback capability is substantially the same as the standard's system-versioned tables. Some key differences are: • In Oracle you do not need to designate particular tables for journaling; all tables are journaled. • In Oracle, LOB columns need to be individually designated for journaling, because of the potential for large amounts of data. The standard has no analogous provision. • In Oracle you need a privilege in order to read historical data. • In the standard, journaled tables have columns to record the start and end timestamps for the row. In Oracle, this is provided through pseudocolumns.

Table C-2 (Cont.) Oracle Support for Optional Features of SQL/Foundation

Feature ID	Feature Support
T181, Application-time period tables	Oracle supports the following elements of this feature: • Application-time period definition during `CREATE TABLE` • Adding and dropping an application-time period definition using `ALTER TABLE` with a minor syntactic difference: Oracle requires parentheses around the period specification; the standard does not support parentheses in this position. Oracle extends this feature: • With the ability to have more than one application-time period per table. • By making the start time and end time columns optional. In this case, Oracle will create these columns implicitly. • By allowing `NULL` for the start time column to indicate that the row is considered valid for any point in time before the value of the end time column. • By allowing `NULL` for the end time column to indicate that the row is considered valid for any point in time on or after the value of the start time column. • By querying an application-time period table using the flashback query options `VERSIONS PERIOD FOR` and `AS OF PERIOD FOR`.
T201, Comparable data types for referential constraints	Oracle fully supports this feature.
T211, Basic trigger capability	Oracle's triggers differ from the standard as follows: • Oracle does not provide the optional syntax `FOR EACH STATEMENT` for the default case, the statement trigger. • Oracle does not support `OLD TABLE` and `NEW TABLE`; the transition tables specified in the standard (the multiset of before and after images of affected rows) are not available. • The trigger body is written in PL/SQL, which is functionally equivalent to the standard's procedural language PSM, but not the same. • In the trigger body, the new and old transition variables are referenced beginning with a colon. • Oracle's row triggers are executed as the row is processed, instead of buffering them and executing all of them after processing all rows. The standard's semantics are deterministic, but Oracle's in-flight row triggers are more performant. • Oracle's before-row and before-statement triggers can perform DML statements, which is forbidden in the standard. However, Oracle's after-row statements cannot perform DML, while it is permitted in the standard. • When multiple triggers apply, the standard says they are executed in order of definition. In Oracle the execution order is nondeterministic, unless specified using `FOLLOWS`. • Oracle uses the system privileges `CREATE TRIGGER` and `CREATE ANY TRIGGER` to regulate creation of triggers, instead of the standard's `TRIGGER` privilege, which is a table privilege.
T212, Enhanced trigger capability	This feature permits statements triggers, which Oracle supports, as described for feature T211, Basic trigger capability.

Table C-2 (Cont.) Oracle Support for Optional Features of SQL/Foundation

Feature ID	Feature Support
T213, INSTEAD OF triggers	Oracle supports INSTEAD OF triggers on views, with syntax and semantics agreeing with the standard except as noted for feature T211, Basic trigger capability. Oracle permits an INSTEAD OF trigger on a view that specified WITH CHECK OPTION, unlike the standard.
T241, START TRANSACTION statement	Oracle's SET TRANSACTION statement starts a transaction making it equivalent to the standard's START TRANSACTION rather than the standard's SET TRANSACTION. Oracle's READ ONLY transactions are at SERIALIZABLE isolation level.
T271, Savepoints	Oracle supports this feature, except: • Oracle does not support RELEASE SAVEPOINT. • Oracle does not support savepoint levels.
T285, Enhanced derived column names	This feature pertains only to derived columns in a SELECT list with no column alias and consisting of a SQL parameter reference. In that case, the column name defaults to the parameter name, the same as in the standard.
T323, Explicit security for external routines	The Oracle syntax AUTHID { CURRENT USER | DEFINER } when used when creating an external function, procedure, or package is equivalent to the standard's EXTERNAL SECURITY { DEFINER | INVOKER }.
T324, Explicit security for SQL routines	Oracle's syntax AUTHID { CURRENT USER | DEFINER } when used when creating a PL/SQL function, procedure, or package is equivalent to the standard's SQL SECURITY { DEFINER | INVOKER }.
T325, Qualified SQL parameter reference	PL/SQL supports the use of a routine name to qualify a parameter name.
T326, Table functions	Oracle provides equivalents for the following elements of this feature: • <multiset value constructor by query> is supported using CAST (MULTISET (<*query expression*>) AS <*nested table type*>) • <table function derived table> is supported using the TABLE operator in the FROM clause with a varray or nested table as the argument • <collection value expression> is equivalent to an Oracle expression resulting in a varray or nested table • <returns table type> is equivalent to a PL/SQL function that returns a nested table
T331, Basic roles	Oracle supports this feature, except for REVOKE ADMIN OPTION FOR <role name>.
T341, Overloading of SQL-invoked functions and procedures	Oracle supports overloading of functions and procedures. However, the rules for handling certain data type combinations are not the same as the standard. For example, the standard permits the coexistence of two functions of the same name differing only in the numeric types of the arguments, whereas Oracle does not permit this.
T351, Bracketed comments	Oracle fully supports this feature.
T431, Extended grouping capabilities	Oracle fully supports this feature.
T432, Nested and concatenated GROUPING SETS	Oracle supports concatenated GROUPING SETS, but not nested GROUPING SETS.

Table C-2 (Cont.) Oracle Support for Optional Features of SQL/Foundation

Feature ID	Feature Support
T433, Multiargument function GROUPING	The Oracle GROUP_ID function can be used to conveniently distinguish groups in a grouped query, serving the same purpose as the standard multiargument GROUPING function.
T441, ABS and MOD functions	Oracle supports the ABS function. Oracle's MOD function is similar to the standard, though the behavior is different if the two arguments are of opposite sign.
T471, Result sets return value	PL/SQL ref cursors provide all the functionality of the standard's result set cursors.
T491, LATERAL derived tables	Oracle fully supports this feature.
T501, Enhanced EXISTS predicate	Oracle fully supports this feature.
T511, Transaction counts	Oracle supports the count of transactions committed and rolled back via the system views V$STATNAME and V$SESSTAT.
T521, Named arguments in CALL statement	Oracle fully supports this feature.
T522, Default values for IN parameters of SQL-invoked procedures	Oracle fully supports this feature.
T524, Named arguments in routine invocations other than a CALL statement	Oracle fully supports this feature.
T525, Default values for parameters of SQL-invoked functions	Oracle fully supports this feature.
T571, Array-returning external SQL-invoked function	Oracle table functions returning a varray can be defined in external programming languages. When declaring such functions in SQL, use the CREATE FUNCTION command with the PIPELINED USING clause.
T572, Multiset-returning external SQL-invoked function	Oracle table functions returning a nested table can be defined in external programming languages. When declaring such functions in SQL, use the CREATE FUNCTION command with the PIPELINED USING clause. In the body of the function, use the OCITable interface. The function must be invoked within the TABLE operator in the FROM clause.
T581, Regular expressions substring functions	Oracle provides the REGEXP_SUBSTR function to perform substring operations using regular expression matching.
T591, UNIQUE constraints of possibly null columns	Oracle permits a UNIQUE constraint on one or more nullable columns. If the UNIQUE constraint is on a single column, then the semantics are the same as the standard (the constraint permits any number of rows that are null in the designated column). If the UNIQUE constraint is on two or more columns, then the semantics are nonstandard. Oracle permits any number of rows that are null in all the designated columns. Unlike the standard, if a row is non-null in at least one of the designated columns, then another row having the same values in the non-null columns of the constraint is a constraint violation and not permitted.
T611, Elementary OLAP operations	Oracle fully supports this feature, except that DISTINCT is only supported in conjunction with window partitioning but not with window framing.

Table C-2 (Cont.) Oracle Support for Optional Features of SQL/Foundation

Feature ID	Feature Support
T612, Advanced OLAP operations	Oracle supports the following elements of this feature: `PERCENT_RANK`, `CUME_DIST`, `WIDTH_BUCKET`, hypothetical set functions, `PERCENTILE_CONT`, `PERCENTILE_DISC`, and `ROW_NUMBER`. Oracle does not support the following element of this feature: • `ROW_NUMBER` without `ORDER BY`
T613, Sampling	Oracle uses the keyword `SAMPLE` instead of the standard's keyword, `TABLESAMPLE`. Oracle uses the keyword `BLOCK` instead of the standard's keyword, `SYSTEM`. Oracle uses the absence of the keyword `BLOCK` to indicate a Bernoulli sampling of rows, indicated in the standard by the keyword `BERNOULLI`. Oracle does not support sampling of derived tables or views that are not key-preserving. Oracle does not permit sampling in a subquery of a `DELETE`, `UPDATE` or `MERGE` statement.
T614, `NTILE` function	Oracle fully supports this feature.
T615, `LEAD` and `LAG` functions	Oracle fully supports this feature.
T616, Null treatment option for `LEAD` and `LAG` functions	Oracle fully supports this feature.
T617, `FIRST_VALUE` and `LAST_VALUE` functions	Oracle fully supports this feature.
T618, `NTH_VALUE` function	Oracle fully supports this feature.
T621, Enhanced numeric functions	Oracle fully supports this feature, except for the alternate spelling `CEILING` of the `CEIL` function.
T622, Trigonometric functions	Oracle fully supports this feature.
T623, General logarithm function	Oracle fully supports this feature.
T625, `LISTAGG`	Oracle fully supports this feature.
T641, Multiple column assignment	The standard syntax to assign to multiple columns is supported if the assignment source is a subquery.
T652, SQL-dynamic statements in SQL routines.	PL/SQL supports dynamic SQL.
T654, SQL-dynamic statements in external routines	Oracle supports dynamic SQL in embedded C, which may be used to create an external routine.
T655, Cyclically dependent routines	PL/SQL supports recursion.
T811, Basic SQL/JSON constructor functions	Oracle fully supports this feature, except for the `JSON_ARRAY` constructor by query.
T812, SQL/JSON: `JSON_OBJECTAGG`	Oracle fully supports this feature.
T813, SQL/JSON: `JSON_ARRAYAGG` with `ORDER BY`	Oracle fully supports this feature.
T821, Basic SQL/JSON query operators	Oracle fully supports this feature.
T822, SQL/JSON: `IS JSON WITH UNIQUE KEYS` predicate	Oracle fully supports this feature.

Table C-2 (Cont.) Oracle Support for Optional Features of SQL/Foundation

Feature ID	Feature Support
T823, SQL/JSON: `PASSING` clause	Oracle supports the `PASSING` clause in `JSON_EXISTS`.
T825, SQL/JSON: `ON EMPTY` and `ON ERROR` clauses	Oracle fully supports this feature, except that: • The `ON ERROR` clause for `JSON_EXISTS` does not support `UNKNOWN`. • `JSON_TABLE` does not support a column-level `ON EMPTY` clause.
T828, `JSON_QUERY`	Oracle fully supports this feature.
T829, `JSON_QUERY`: array wrapper options	Oracle fully supports this feature.
T832, SQL/JSON path language: item method	Oracle fully supports the following item methods: • `abs` • `ceiling` • `double` • `floor` Oracle provides the following comparable support: • `date` and `timestamp` are comparable to the standard's `datetime` Oracle extends this feature by supporting the following item methods: • `length` • `lower` • `number` • `string` • `upper`
T833, SQL/JSON path language: multiple subscripts	Oracle fully supports this feature, except that subscripts have to be specified in strictly monotonically increasing order.
T834, SQL/JSON path language: wildcard member accessor	Oracle fully supports this feature.
T835, SQL/JSON path language: filter expression	Oracle supports the filter expression as the last step of the SQL/JSON path expression in `JSON_EXISTS`.
T839, Formatted cast of datetimes to/from character strings	Oracle supports this feature with a minor syntactic difference: Oracle uses a comma instead of the keyword `FORMAT`.

Oracle Compliance with SQL/CLI

The Oracle ODBC driver conforms to SQL/CLI.

Oracle Compliance with SQL/PSM

Oracle PL/SQL provides functionality equivalent to SQL/PSM, with minor syntactic differences, such as the spelling or arrangement of keywords.

Oracle Compliance with SQL/MED

Oracle does not comply with SQL/MED.

Oracle Compliance with SQL/OLB

Oracle SQLJ conforms to SQL/OLB:1999 and not yet to SQL/OLB:2016.

Oracle Compliance with SQL/JRT

Oracle fully supports stored routines and SQL types implemented in Java(TM). Oracle provides equivalent support for the creation and maintenance of such types and procedures. Oracle's capabilities are in general a superset of the functionality defined by the standard.

Oracle Compliance with SQL/XML

The XML data type in the standard is `XML`. The Oracle equivalent data type is `XMLType`. A feature of the standard is considered to be fully supported if the only difference between Oracle and the standard is the spelling of the data type name.

Table C-3 describes Oracle's support for the features of SQL/XML.

Table C-3 Oracle Support for Features of SQL/XML

Feature ID	Feature Support
X010, XML type	Oracle fully supports this feature.
X011, Arrays of XML types	Oracle supports this feature using named array types
X012, Multisets of XML type	The Oracle equivalent of a multiset of XML type is a nested table with a single column of XML type.
X013, Distinct types of XML	A distinct type can be emulated using an object type with a single attribute.
X014, Attributes of XML type	In Oracle, attributes of object types may be of type `XMLType`, but the syntax for creating object types is nonstandard.
X015, Fields of XML type	Oracle object types may be used instead of row types; Oracle supports object types with attributes of `XMLType`.
X016, Persistent XML values	Oracle fully supports this feature.
X020, `XMLConcat`	Oracle fully supports this feature.
X025, `XMLCast`	Oracle supports this feature, with the following restrictions: • The source expression must be of `XMLType` and the target data type may not be `XMLType`. (Since Oracle has only one XML type, there is no need to cast from XML to XML.) • Oracle does not support <XML passing mechanism>; the behavior is the same as `BY VALUE` in the standard. Oracle extends this feature with the ability to cast to type `REF XMLTYPE`.
X031, `XMLElement`	Oracle fully supports this feature.
X032, `XMLForest`	Oracle fully supports this feature.

Table C-3 (Cont.) Oracle Support for Features of SQL/XML

Feature ID	Feature Support
X034, `XMLAgg`	Oracle fully supports this feature.
X035, `XMLAgg`: `ORDER BY` option	Oracle fully supports this feature.
X036, `XMLComment`	Oracle fully supports this feature.
X036, `XMLPi`	Oracle fully supports this feature.
X038, `XMLText`	The Oracle `XMLCData` function may be used to create a text node.
X040, Basic table mapping	Oracle table mappings are available through a Java interface and through a package. Oracle table mappings have been generalized to map queries and not just tables. To map only a table: `SELECT * FROM` *`table_name`*. This provides support for the following elements of this feature: • X041, Basic table mapping: null absent • X042, Basic table mapping: null as nil • X043, Basic table mapping: table as forest • X044, Basic table mapping: table as element • X045, Basic table mapping: with target namespace • X046, Basic table mapping: data mapping • X047, Basic table mapping: metadata mapping • X049, Basic table mapping: hex encoding Oracle does not support the following element of this feature: • X048, Basic table mapping: base64 encoding
X041, Basic table mapping: null absent	See X040.
X042, Basic table mapping: null as nil	See X040.
X043, Basic table mapping: table as forest	See X040.
X044, Basic table mapping: table as element	See X040.
X045, Basic table mapping: with target namespace	See X040.
X046, Basic table mapping: data mapping	See X040.
X047, Basic table mapping: metadata mapping	See X040.
X049, Basic table mapping: hex encoding	See X040.
X060, `XMLParse`: Character string input and `CONTENT` option	Oracle does not support the {`PRESERVE` | `STRIP`} `WHITESPACE` syntax. The behavior is always `STRIP WHITESPACE`.
X061, `XMLParse`: Character string input and `DOCUMENT` option	Oracle does not support the {`PRESERVE` | `STRIP`} `WHITESPACE` syntax. The behavior is always `STRIP WHITESPACE`.
X069, `XMLSERIALIZE`: `INDENT`	Oracle extends this feature with the ability to specify an indent size.

Table C-3 (Cont.) Oracle Support for Features of SQL/XML

Feature ID	Feature Support
X070, `XMLSerialize`: Character string serialization and `CONTENT` option	Oracle supports this feature, with this restriction: • In the standard, the choice of `DOCUMENT` or `CONTENT` is optional; in Oracle, you must specify one of these. Oracle extends this feature as follows: the standard requires a target data type; Oracle defaults to `CLOB`.
X071, `XMLSerialize`: Character string serialization and `DOCUMENT` option	Oracle fully supports this feature.
X072, `XMLSerialize`: Character string serialization	Oracle fully supports this feature.
X073, `XMLSerialize`: `BLOB` serialization and `CONTENT` option	Oracle fully supports this feature.
X074, `XMLSerialize`: `BLOB` serialization and `DOCUMENT` option	Oracle fully supports this feature.
X075, `XMLSerialize`: `BLOB` serialization	Oracle fully supports this feature.
X076, `XMLSerialize`: `VERSION` option	Oracle fully supports this feature.
X077, `XMLSerialize`: explicit `ENCODING` option	Oracle fully supports this feature.
X080, Namespaces in XML publishing	In the Oracle implementation of `XMLElement`, `XMLAttributes` are used to define namespaces (`XMLNamespaces` is not implemented). However, `XMLAttributes` is not supported for `XMLForest`.
X086, XML namespace declarations in `XMLTable`	Oracle fully supports this feature.
X090, XML document predicate	In Oracle, you can test whether an XML value is a document by using the `ISFRAGMENT` method.
X096, `XMLExists`	Oracle fully supports this feature, with this exception: Oracle only supports passing by value, so the keywords `BY VALUE` are optional at the beginning of the `PASSING` clause, and not supported on individual arguments.
X120, XML parameters in SQL routines	Oracle fully supports this feature.
X121, XML parameters in external routines	Oracle supports XML values passed to external routines using a non-standard interface.
X141, `IS VALID` predicate: data drive case	The `XMLISVALID` method is equivalent to the `IS VALID` predicate, and supports the data-driven case.
X142, `IS VALID` predicate: `ACCORDING TO` clause	The `XMLISVALID` method is equivalent to the `IS VALID` predicate, and includes the equivalent of the `ACCORDING TO` clause.
X143, `IS VALID` predicate: `ELEMENT` clause	The `XMLISVALID` method is equivalent to the `IS VALID` predicate, and includes the equivalent of the `ELEMENT` clause.
X144, `IS VALID` predicate: schema location	The `XMLISVALID` method is equivalent to the `IS VALID` predicate, and supports the specification of a schema location for a registered XML Schema.

Table C-3 (Cont.) Oracle Support for Features of SQL/XML

Feature ID	Feature Support
X145, IS VALID predicate outside check constraints	The XMLISVALID method is equivalent to the IS VALID predicate, and may be used outside check constraints.
X151, IS VALID predicate with DOCUMENT option	The XMLISVALID method is equivalent to the IS VALID predicate, and performs validation equivalent to the DOCUMENT clause. (XMLISVALID does not support "content" validation.)
X156, IS VALID predicate: optional NAMESPACE with ELEMENT clause	The XMLISVALID method is equivalent to the IS VALID predicate, and may be used to validate against an element in any namespace.
X157, IS VALID predicate: NO NAMESPACE with ELEMENT clause	The XMLISVALID method is equivalent to the IS VALID predicate, and may be used to validate against an element in the "no name" namespace.
X160, Basic Information Schema for registered XML Schemas	The Oracle static data dictionary view ALL_XML_SCHEMAS provides a list of the registered XML schemas that are accessible to the current user. The ALL_XML_SCHEMAS.SCHEMA_URL column corresponds to the standard XML_SCHEMAS.XML_SCHEMA_LOCATION column. The target namespace of the registered XML Schemas can be learned by examining ALL_XML_SCHEMAS.SCHEMA. Oracle has no equivalents for the other columns of the standard's XML_SCHEMAS.
X161, Advanced Information Schema for registered XML Schemas	Oracle does not have static data dictionary views corresponding to XML_SCHEMA_NAMESPACES and XML_SCHEMA_ELEMENTS in the standard. However, all the information about registered XML Schemas may be learned by examining the actual XML Schema, which is found in the ALL_XML_SCHEMAS.SCHEMA column. This may also be examined to learn whether a registered XML Schema is nondeterministic, and which of its namespaces and elements are nondeterministic.
X191, XML(DOCUMENT (XMLSCHEMA)) type	Oracle does not support this syntax. However, a column of a table can be constrained by a registered XML Schema, in which case all values of the column will be of XML(DOCUMENT(XMLSCHEMA)) type.
X200, XMLQuery	Oracle fully supports the following elements of this feature: • X201, XMLQuery: RETURNING CONTENT • X203, XMLQuery: passing a context item • X204, XMLQuery: initializing an XQuery variable • X206, XMLQuery: NULL ON EMPTY option Oracle only supports passing by value, so the keywords BY VALUE are optional at the beginning of the PASSING clause, and not supported on individual arguments.
X201, XMLQuery: RETURNING CONTENT	See X200.
X203, XMLQuery: passing a context item	See X200.
X204, XMLQuery: initializing an XQuery variable	See X200.
X206, XMLQuery: NULL ON EMPTY option	See X200.

Table C-3 (Cont.) Oracle Support for Features of SQL/XML

Feature ID	Feature Support
X221, XML passing mechanism BY VALUE	Oracle supports the BY VALUE clause in XMLQuery, XMLTable and XMLExists. In these, BY VALUE is supported as optional syntax at the beginning of an argument list, but not as a modifier on an individual argument or column.
X232, XML(CONTENT(ANY)) type	Oracle does not support this syntax as a type modifier, but the Oracle XMLType supports this data type for transient values. Persistent values are of type XML(DOCUMENT(ANY)), which is a subset of XML(CONTENT(ANY)).
X241, RETURNING CONTENT in XML publishing	Oracle does not support this syntax. In Oracle, the behavior of the publishing functions (XMLAgg, XMLComment, XMLConcat, XMLElement, XMLForest, and XMLPi) is always RETURNING CONTENT.
X251, Persistent XML values of XML(DOCUMENT(UNTYPED)) type	Oracle fully supports this feature.
X252, Persistent values of type XML(DOCUMENT(ANY))	Oracle fully supports this feature.
X256, Persistent values of XML(DOCUMENT(XMLSCHEMA)) type	Oracle fully supports this feature.
X260, XML type, ELEMENT clause	Oracle does not support this syntax. However, a column of a table may be constrained by a top-level element in a registered XML Schema.
X263, XML type: NO NAMESPACE with ELEMENT clause	Oracle does not support this syntax. However, a column of a table may be constrained by a top-level element in the "no name" namespace of a registered XML Schema.
X264, XML type: schema location	Oracle does not support this syntax. However, a column of a table may be constrained by a registered XML Schema that is identified by a schema location.
X271, XMLValidate: data driven case	The SCHEMAVALIDATE method is equivalent to XMLValidate, and supports the data-driven case.
X272, XMLValidate: ACCORDING TO clause	The SCHEMAVALIDATE method is equivalent to XMLValidate, and may be used to specify a particular registered XML Schema.
X273, XMLValidate: ELEMENT clause	The SCHEMAVALIDATE method is equivalent to XMLValidate, and may be used to specify a particular element of a particular registered XML Schema.
X274, XMLValidate: schema location	The SCHEMAVALIDATE method is equivalent to XMLValidate, and may be used to specify a particular registered XML Schema by its schema location URL.
X281, XMLValidate with DOCUMENT option	The SCHEMAVALIDATE method is equivalent to XMLValidate. SCHEMAVALIDATE performs validation only of XML documents (not content).
X286, XMLValidate: NO NAMESPACE with ELEMENT clause	The SCHEMAVALIDATE method is equivalent to XMLValidate, and may be used to specify a particular element in the "no name" namespace of a particular registered XML Schema.

Table C-3 (Cont.) Oracle Support for Features of SQL/XML

Feature ID	Feature Support
X300, `XMLTable`	Oracle does not support reverse axes in the column path expressions. Aside from that restriction, Oracle fully supports the following elements of this feature: • X086, XML namespace declarations in `XMLTable` • X302, `XMLTable` with ordinality column • X303, `XMLTable`: column default option • X304, `XMLTable`: passing a context item • X305, `XMLTable`: initializing an XQuery variable Oracle only supports passing by value, so the keywords `BY VALUE` are optional at the beginning of the `PASSING` clause, and not supported on individual arguments.
X302, `XMLTable` with ordinality column	See X300.
X303, `XMLTable`: column default option	See X300.
X304, `XMLTable`: passing a context item	See X300.
X305, `XMLTable`: initializing an XQuery variable	See X300.

Oracle Compliance with SQL/MDA

Oracle does not comply with SQL/MDA.

Oracle Compliance with SQL/PGQ

Table Table C-4 describes Oracle's support for the features of SQL/PGQ.

Table C-4 Oracle Support for Features of SQL/PGQ

Feature ID	Feature Support
G000, Graph pattern	Oracle fully supports this feature.
G001, Repeatable-elements match mode	Oracle fully supports this feature.
G008, Graph pattern WHERE clause	Oracle fully supports this feature.
G034, Path concatenation	Oracle fully supports this feature.
G035, Quantified paths	Oracle fully supports this feature.
G036, Quantified edges	Oracle fully supports this feature.
G037, Questioned paths	Oracle fully supports this feature.
G040, Vertex pattern	Oracle fully supports this feature.
G042, Basic full edge patterns	Oracle fully supports this feature.
G044, Basic abbreviated edge patterns	Oracle fully supports this feature.
G060, Bounded graph pattern quantifiers	Oracle fully supports this feature.

Table C-4 (Cont.) Oracle Support for Features of SQL/PGQ

Feature ID	Feature Support
G070, Label expression: label disjunction	Oracle fully supports this feature.
G071, Label expression: label conjunction	Oracle provides equivalent functionality using label expressions on different graph patterns with the same graph element variable name.
G073, Label expression: individual label name	Oracle fully supports this feature.
G090, Property reference	Oracle fully supports this feature.
G100, `ELEMENT_ID` function	Oracle provides equivalent functionality using `EDGE_ID` and `VERTEX_ID` operators.
G112, `IS SOURCE` and `IS DESTINATION` predicate	Oracle fully supports this feature.
G114, `SAME` predicate	Oracle provides equivalent functionality using `VERTEX_EQUAL` and `EDGE_EQUAL` predicates.
G120, Within-match aggregates	Oracle fully supports this feature.
G900, `GRAPH_TABLE`	Oracle fully supports this feature.
G904, All properties reference	Oracle supports this feature in the `COLUMNS` clause.
G920, DDL-based SQL-property graphs	Oracle fully supports this feature with the following exception: the keyword `RESTRICT` is not supported for the `DROP PROPERTY GRAPH` statement.
G924, Explicit key clause for element tables	Oracle fully supports this feature.
G925, Explicit label and properties clause for element tables	Oracle fully supports this feature.
G926, More than one label for vertex tables	Oracle fully supports this feature.
G927, More than one label for edge tables	Oracle fully supports this feature.
G928, Value expressions as properties and renaming of properties	Oracle fully supports this feature.
G929, Labels and properties: `EXCEPT` list	Oracle fully supports this feature.
G940, Multi-sourced and multi-destined edges	Oracle fully supports this feature.
G941, Implicit removal of incomplete edges	Oracle fully supports this feature.

Oracle Compliance with FIPS 127-2

Oracle complied fully with last Federal Information Processing Standard (FIPS), which was FIPS PUB 127-2. That standard is no longer published. However, for users whose applications depend on information about the sizes of some database constructs that were defined in FIPS 127-2, the details of our compliance are listed in Table C-5.

Table C-5 Sizing for Database Constructs

Database Constructs	FIPS	Oracle Database
Length of an identifier (in bytes)	18	128
Length of `CHARACTER` data type (in bytes)	240	2,000
Decimal precision of `NUMERIC` data type	15	38

Table C-5 (Cont.) Sizing for Database Constructs

Database Constructs	FIPS	Oracle Database
Decimal precision of `DECIMAL` data type	15	38
Decimal precision of `INTEGER` data type	9	38
Decimal precision of `SMALLINT` data type	4	38
Binary precision of `FLOAT` data type	20	126
Binary precision of `REAL` data type	20	63
Binary precision of `DOUBLE PRECISION` data type	30	126
Columns in a table	100	1,000
Values in an `INSERT` statement	100	1,000
SET clauses in an `UPDATE` statement (Note 1)	20	1,000
Length of a row (Note2, Note 3)	2,000	2,000,000
Columns in a `UNIQUE` constraint	6	32
Length of a `UNIQUE` constraint (Note 2)	120	(Note 4)
Length of foreign key column list (Note 2)	120	(Note 4)
Columns in a `GROUP BY` clause	6	255 (Note 5)
Length of `GROUP BY` column list	120	(Note 5)
Sort specifications in `ORDER BY` clause	6	255 (Note 5)
Length of `ORDER BY` column list	120	(Note 5)
Columns in a referential integrity constraint	6	32
Tables referenced in a SQL statement	15	No limit
Cursors simultaneously open	10	(Note 6)
Items in a `SELECT` list	100	1,000

Note 1: The number of `SET` clauses in an `UPDATE` statement refers to the number items separated by commas following the `SET` keyword.

Note 2: The FIPS PUB defines the length of a collection of columns to be the sum of: twice the number of columns, the length of each character column in bytes, decimal precision plus 1 of each exact numeric column, binary precision divided by 4 plus 1 of each approximate numeric column.

Note 3: The Oracle limit for the maximum row length is based on the maximum length of a row containing a `LONG` value of length 2 gigabytes and 999 `VARCHAR2` values, each of length 4000 bytes: 2(254) + 231 + (999(4000)).

Note 4: The Oracle limit for a `UNIQUE` key is half the size of an Oracle data block (specified by the initialization parameter `DB_BLOCK_SIZE`) minus some overhead.

Note 5: Oracle places no limit on the number of columns in a `GROUP BY` clause or the number of sort specifications in an `ORDER BY` clause. However, the sum of the sizes of all the expressions in either a `GROUP BY` clause or an `ORDER BY` clause is limited to the size of an Oracle data block (specified by the initialization parameter `DB_BLOCK_SIZE`) minus some overhead.

Note 6: The Oracle limit for the number of cursors simultaneously opened is specified by the initialization parameter `OPEN_CURSORS`. The maximum value of this parameter depends on the memory available on your operating system and exceeds 100 in all cases.

Oracle Extensions to Standard SQL

Oracle supports numerous features that extend beyond standard SQL. If you are concerned with the portability of your applications to other implementations of SQL, then use Oracle's FIPS Flagger to help identify the use of Oracle extensions to Entry SQL-92 in your embedded SQL programs. The FIPS Flagger is part of the Oracle precompilers and the SQL*Module compiler. The FIPS Flagger can also be enabled in SQL*Plus by using `ALTER SESSION SET FLAGGER = ENTRY`. While SQL-92 has been superseded by SQL:2016, there has been no conformance testing authority for any version of SQL since SQL-92; hence, Entry SQL-92 offers you the most assurance of portability.

> **See Also:**
>
> *Pro*COBOL Programmer's Guide* and *Pro*C/C++ Programmer's Guide* for information on how to use the FIPS Flagger

Oracle Compliance with Older Standards

This release of Oracle Database conforms to SQL:2016, the most recent edition of the SQL standard when this guide was published, as itemized in preceding sections of this appendix. Oracle does not formally claim that this release of the database conforms to SQL-92—and in particular, to SQL-92 Entry Level—or to SQL:1999, because those standards have been superseded by SQL:2016. Some, mostly minor, changes between editions of the SQL standard might affect applications. The SQL standard, or a reference discussing that standard, can be consulted to determine the details of any incompatibilities that have been introduced. One important source is Annex E of SQL/Foundation:1999, SQL/Foundation:2003, SQL/Foundation:2008, SQL/Foundation:2011, and SQL/Foundation:2016.

In some cases, this release of Oracle Database might continue to recognize constructs from older editions of SQL. Such recognition is often allowed as a valid vendor extension. It is the general policy of Oracle to keep incompatibilities between versions of the database as few as possible. This policy extends to retention of older forms when that is feasible. In any case, the differences between older SQL and SQL:2016 (as noted above) are relatively inconsequential.

Character Set Support

Oracle supports most national, international, and vendor-specific encoded character set standards. A complete list of character sets supported by Oracle appears in *Oracle Database Globalization Support Guide*.

Unicode is a universal encoded character set that lets you store information from any language using a single character set. Unicode is required by modern standards such as XML, Java, JavaScript, and LDAP. Unicode is compliant with ISO/IEC standard 10646. For information on ISO standards, visit the Web site of the International Organization for Standardization:

http://www.iso.ch/

Oracle Database Release 23 complies with version 15.0 of the Unicode Standard. For up-to-date information on the Unicode Standard, visit the Web site of the Unicode Consortium:

http://www.unicode.org

Oracle supports the UTF-8 encoding scheme of the Unicode Standard through the AL32UTF8 character set, the UTF-16BE encoding scheme through the AL16UTF16 character set, and the UTF-16LE encoding scheme through the AL16UTF16LE character set. AL32UTF8 is valid as the client and database character set on ASCII-based platforms. AL16UTF16 is valid as the national (NCHAR) character set on all platforms. AL16UTF16LE is not valid as the client, database, or national character set.

Oracle implements two deprecated Unicode compatibility encoding forms: CESU-8 through the UTF8 character set and UTF-EBCDIC through the UTFE character set. The UTF8 and UTFE character sets are not guaranteed to include updates to the Unicode standard beyond version 3.0. UTF8 is valid as the client and database character set on ASCII-based platforms and as the national (NCHAR) character set on all platforms. UTFE is valid as the database character set on EBCDIC-based platforms.

All mentioned Oracle character sets are supported in conversion functions.

Oracle recommends that databases on ASCII-based platforms are created with the AL32UTF8 character set and the AL16UTF16 national (NCHAR) character set. Oracle recommends that you avoid the use of the NCHAR data types and the associated national character set as they are not supported by some RDBMS components, such as Oracle Text and Oracle XDB.

> **See Also:**
>
> *Oracle Database Globalization Support Guide* for details on Oracle character set support