Annex A

(informative)

SQL Conformance Summary

The contents of this Annex summarizes all Conformance Rules, ordered by Feature ID and by Subclause.

1) Specifications for Feature B011, "Embedded Ada":
   a) Subclause 20.3, "<embedded SQL Ada program>":
      i) Without Feature B011, "Embedded Ada", conforming SQL language shall not contain an <embedded SQL Ada program>.
2) Specifications for Feature B012, "Embedded C":
   a) Subclause 20.4, "<embedded SQL C program>":
      i) Without Feature B012, "Embedded C", conforming SQL language shall not contain an <embedded SQL C program>.
3) Specifications for Feature B013, "Embedded COBOL":
   a) Subclause 20.5, "<embedded SQL COBOL program>":
      i) Without Feature B013, "Embedded COBOL", conforming SQL language shall not contain an <embedded SQL COBOL program>.
4) Specifications for Feature B014, "Embedded Fortran":
   a) Subclause 20.6, "<embedded SQL Fortran program>":
      i) Without Feature B014, "Embedded Fortran", conforming SQL language shall not contain an <embedded SQL Fortran program>.
5) Specifications for Feature B015, "Embedded MUMPS":
   a) Subclause 20.7, "<embedded SQL MUMPS program>":
      i) Without Feature B015, "Embedded MUMPS", conforming SQL language shall not contain an <embedded SQL MUMPS program>.
6) Specifications for Feature B016, "Embedded Pascal":
   a) Subclause 20.8, "<embedded SQL Pascal program>":
      i) Without Feature B016, "Embedded Pascal", conforming SQL language shall not contain an <embedded SQL Pascal program>.
7) Specifications for Feature B017, "Embedded PL/I":

a) Subclause 20.9, "<embedded SQL PL/I program>":

i) Without Feature B017, "Embedded PL/I", conforming SQL language shall not contain an <embedded SQL PL/I program>.

8) Specifications for Feature B021, "Direct SQL":

a) Subclause 21.1, "<direct SQL statement>":

i) Without Feature B021, "Direct SQL", conforming SQL language shall not contain a <direct SQL statement>.

9) Specifications for Feature B031, "Basic dynamic SQL":

a) Subclause 5.4, "Names and identifiers":

i) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain an <SQL statement name>.

ii) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain <dynamic cursor name>.

iii) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain a <descriptor name>.

b) Subclause 6.4, "<value specification> and <target specification>":

i) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain a <general value specification> that contains a <dynamic parameter specification>.

c) Subclause 19.2, "<allocate descriptor statement>":

i) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain an <allocate descriptor statement>.

d) Subclause 19.3, "<deallocate descriptor statement>":

i) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain a <deallocate descriptor statement>.

e) Subclause 19.4, "<get descriptor statement>":

i) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain a <get descriptor statement>.

f) Subclause 19.5, "<set descriptor statement>":

i) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain a <set descriptor statement>.

g) Subclause 19.6, "<prepare statement>":

i) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain a <prepare statement>.

h) Subclause 19.9, "<describe statement>":

i) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain a <describe output statement>.

i) Subclause 19.10, "<input using clause>":

i) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain an <input using clause>.

j) Subclause 19.11, "<output using clause>":

i) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain an <output using clause>.

k) Subclause 19.12, "<execute statement>":

i) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain an <execute statement>.

l) Subclause 19.13, "<execute immediate statement>":

i) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain an <execute immediate statement>.

m) Subclause 19.14, "<dynamic declare cursor>":

i) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain a <dynamic declare cursor>.

n) Subclause 19.16, "<dynamic open statement>":

i) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain a <dynamic open statement>.

o) Subclause 19.17, "<dynamic fetch statement>":

i) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain a <dynamic fetch statement>.

p) Subclause 19.18, "<dynamic single row select statement>":

i) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain a <dynamic single row select statement>.

q) Subclause 19.19, "<dynamic close statement>":

i) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain a <dynamic close statement>.

r) Subclause 19.20, "<dynamic delete statement: positioned>":

i) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain a <dynamic delete statement: positioned>.

s) Subclause 19.21, "<dynamic update statement: positioned>":

i) Without Feature B031, "Basic dynamic SQL", conforming SQL language shall not contain a <dynamic update statement: positioned>.

10) Specifications for Feature B032, "Extended dynamic SQL":

a) Subclause 5.4, "Names and identifiers":

i) Without Feature B032, "Extended dynamic SQL", conforming SQL language shall not contain a <extended statement name> or <extended cursor name>.

ii) Without Feature B032, "Extended dynamic SQL", conforming SQL language shall not contain a <descriptor name> that is not a <literal>.

b) Subclause 19.2, "<allocate descriptor statement>":

i) Without Feature B032, "Extended dynamic SQL", conforming SQL language shall not contain an <occurrences> that is not a <literal>.

c) Subclause 19.8, "<deallocate prepared statement>":

i) Without Feature B032, "Extended dynamic SQL", conforming SQL language shall not contain a <deallocate prepared statement>.

d) Subclause 19.9, "<describe statement>":

i) Without Feature B032, "Extended dynamic SQL", conforming SQL language shall not contain a <describe input statement>.

e) Subclause 19.12, "<execute statement>":

i) Without Feature B032, "Extended dynamic SQL", conforming SQL language shall not contain a <result using clause>.

f) Subclause 19.15, "<allocate cursor statement>":

i) Without Feature B032, "Extended dynamic SQL", conforming SQL language shall not contain an <allocate cursor statement>.

g) Subclause 19.22, "<preparable dynamic delete statement: positioned>":

i) Without Feature B032, "Extended dynamic SQL", conforming SQL language shall not contain a <preparable dynamic delete statement: positioned>.

h) Subclause 19.23, "<preparable dynamic update statement: positioned>":

i) Without Feature B032, "Extended dynamic SQL", conforming SQL language shall not contain a <preparable dynamic update statement: positioned>.

11) Specifications for Feature B033, "Untyped SQL-invoked function arguments":

a) Subclause 10.4, "<routine invocation>":

i) Without Feature B033, "Untyped SQL-invoked function arguments", conforming SQL language shall not contain a <routine invocation> that is not simply contained in a <call statement> that simply contains an <SQL argument> that is a <dynamic parameter specification>.

12) Specifications for Feature B034, "Dynamic specification of cursor attributes":

a) Subclause 19.6, "<prepare statement>":

i) Without Feature B034, "Dynamic specification of cursor attributes", conforming SQL language shall not contain an <attributes specification>.

13) Specifications for Feature B041, "Extensions to embedded SQL exception declarations":

a) Subclause 20.2, "<embedded exception declaration>":

i) Without Feature B041, "Extensions to embedded SQL exception declarations", conforming SQL language shall not contain an <SQL condition> that contains either SQLSTATE or CONSTRAINT.

14) Specifications for Feature B051, "Enhanced execution rights":

a) Subclause 13.1, "<SQL-client module definition>":

i) Without Feature B051, "Enhanced execution rights", conforming SQL language shall not contain a <module authorization clause> that immediately contains FOR STATIC ONLY or FOR STATIC AND DYNAMIC.

b) Subclause 20.1, "<embedded SQL host program>":

i) Without Feature B051, "Enhanced execution rights", conforming SQL language shall not contain an <embedded authorization declaration>.

15) Specifications for Feature B111, "Module language Ada":

a) Subclause 13.1, "<SQL-client module definition>":

i) Without Feature B111, "Module language Ada", conforming SQL language shall not contain an <SQL-client module definition> that contains a <language clause> that contains ADA.

16) Specifications for Feature B112, "Module language C":

a) Subclause 13.1, "<SQL-client module definition>":

i) Without Feature B112, "Module language C", conforming SQL language shall not contain an <SQL-client module definition> that contains a <language clause> that contains C.

17) Specifications for Feature B113, "Module language COBOL":

a) Subclause 13.1, "<SQL-client module definition>":

i) Without Feature B113, "Module language COBOL", conforming SQL language shall not contain an <SQL-client module definition> that contains a <language clause> that contains COBOL.

18) Specifications for Feature B114, "Module language Fortran":

a) Subclause 13.1, "<SQL-client module definition>":

i) Without Feature B114, "Module language Fortran", conforming SQL language shall not contain an <SQL-client module definition> that contains a <language clause> that contains FORTRAN.

19) Specifications for Feature B115, "Module language MUMPS":

a) Subclause 13.1, "<SQL-client module definition>":

i) Without Feature B115, "Module language MUMPS", conforming SQL language shall not contain an <SQL-client module definition> that contains a <language clause> that contains M.

20) Specifications for Feature B116, "Module language Pascal":

a) Subclause 13.1, "<SQL-client module definition>":

i) Without Feature B116, "Module language Pascal", conforming SQL language shall not contain an <SQL-client module definition> that contains a <language clause> that contains PASCAL.

21) Specifications for Feature B117, "Module language PL/I":

a) Subclause 13.1, "<SQL-client module definition>":

i) Without Feature B117, "Module language PL/I", conforming SQL language shall not contain an <SQL-client module definition> that contains a <language clause> that contains PLI.

22) Specifications for Feature B121, "Routine language Ada":

a) Subclause 11.50, "<SQL-invoked routine>":

i) Without Feature B121, "Routine language Ada", conforming SQL language shall not contain a <routine characteristic> that contains a <language clause> that contains ADA.

23) Specifications for Feature B122, "Routine language C":

a) Subclause 11.50, "<SQL-invoked routine>":

i) Without Feature B122, "Routine language C", conforming SQL language shall not contain a <routine characteristic>that contains a <language clause> that contains C.

24) Specifications for Feature B123, "Routine language COBOL":

a) Subclause 11.50, "<SQL-invoked routine>":

i) Without Feature B123, "Routine language COBOL", conforming SQL language shall not contain a <routine characteristic> that contains a <language clause> that contains COBOL.

25) Specifications for Feature B124, "Routine language Fortran":

a) Subclause 11.50, "<SQL-invoked routine>":

i) Without Feature B124, "Routine language Fortran", conforming SQL language shall not contain a <routine characteristic> that contains a <language clause> that contains FORTRAN.

26) Specifications for Feature B125, "Routine language MUMPS":

a) Subclause 11.50, "<SQL-invoked routine>":

i) Without Feature B125, "Routine language MUMPS", conforming SQL language shall not contain a <routine characteristic> that contains a <language clause> that contains M.

27) Specifications for Feature B126, "Routine language Pascal":

a) Subclause 11.50, "<SQL-invoked routine>":

i) Without Feature B126, "Routine language Pascal", conforming SQL language shall not contain a <routine characteristic> that contains a <language clause> that contains PASCAL.

28) Specifications for Feature B127, "Routine language PL/I":

a) Subclause 11.50, "<SQL-invoked routine>":

i) Without Feature B127, "Routine language PL/I", conforming SQL language shall not contain a <routine characteristic> that contains a <language clause> that contains PLI.

29) Specifications for Feature B128, "Routine language SQL":

a) Subclause 11.50, "<SQL-invoked routine>":

i) Without Feature B128, "Routine language SQL", conforming SQL language shall not contain a <routine characteristic> that contains a <language clause> that contains SQL.

30) Specifications for Feature F032, "CASCADE drop behavior":

a) Subclause 11.21, "<drop table statement>":

i) Without Feature F032, "CASCADE drop behavior", conforming SQL language shall not contain a <drop table statement> that contains <drop behavior> that contains CASCADE.

b) Subclause 11.23, "<drop view statement>":

i) Without Feature F032, "CASCADE drop behavior", conforming SQL language shall not contain a <drop view statement> that contains a <drop behavior> that contains CASCADE.

c) Subclause 11.49, "<drop data type statement>":

i) Without Feature F032, "CASCADE drop behavior", conforming SQL language shall not contain a <drop data type statement> that contains a <drop behavior> that contains CASCADE.

d) Subclause 11.52, "<drop routine statement>":

i) Without Feature F032, "CASCADE drop behavior", conforming SQL language shall not contain a <drop routine statement> that contains a <drop behavior> that contains CASCADE.

31) Specifications for Feature F033, "ALTER TABLE statement: DROP COLUMN clause":

a) Subclause 11.18, "<drop column definition>":

i) Without Feature F033, "ALTER TABLE statement: DROP COLUMN clause", conforming SQL language shall not contain a <drop column definition>.

32) Specifications for Feature F034, "Extended REVOKE statement":

a) Subclause 12.7, "<revoke statement>":

i) Without Feature F034, "Extended REVOKE statement", conforming SQL language shall not contain a <revoke statement> that contains a <drop behavior> that contains CASCADE.

ii) Without Feature F034, "Extended REVOKE statement", conforming SQL language shall not contain a <revoke option extension> that contains GRANT OPTION FOR.

iii) Without Feature F034, "Extended REVOKE statement", conforming SQL language shall not contain a <revoke statement> that contains a <privileges> that contains an <object name> where the owner of the SQL-schema that is specified explicitly or implicitly in the <object name> is not the current authorization identifier.

iv) Without Feature F034, "Extended REVOKE statement", conforming SQL language shall not contain a <revoke statement> such that there exists a privilege descriptor *PD* that satisfies all the following conditions:

1) *PD* identifies the object identified by <object name> simply contained in <privileges> contained in the <revoke statement>.

2) *PD* identifies the <grantee> identified by any <grantee> simply contained in <revoke statement> and that <grantee> does not identify the owner of the SQL-schema that is specified explicitly or implicitly in the <object name> simply contained in <privileges> contained in the <revoke statement>.

3) *PD* identifies the action identified by the <action> simply contained in <privileges> contained in the <revoke statement>.

4) *PD* indicates that the privilege is grantable.

33) Specifications for Feature F052, "Intervals and datetime arithmetic":

a) Subclause 5.3, "<literal>":

i) Without Feature F052, "Intervals and datetime arithmetic", conforming SQL language shall not contain an <interval literal>.

b) Subclause 6.1, "<data type>":

i) Without Feature F052, "Intervals and datetime arithmetic", conforming SQL language shall not contain an <interval type>.

c) Subclause 6.27, "<numeric value function>":

i) Without Feature F052, "Intervals and datetime arithmetic", conforming SQL language shall not contain an <extract expression>.

ii) Without Feature F052, "Intervals and datetime arithmetic", conforming SQL language shall not contain an <extract expression> that specifies a <time zone field>.

d) Subclause 6.30, "<datetime value expression>":

i) Without Feature F052, "Intervals and datetime arithmetic", conforming SQL language shall not contain <datetime value expression> that immediately contains a <plus sign> or a <minus sign>.

e) Subclause 6.32, "<interval value expression>":

i) Without Feature F052, "Intervals and datetime arithmetic", conforming SQL language shall not contain an <interval value expression>.

f) Subclause 6.33, "<interval value function>":

i) Without Feature F052, "Intervals and datetime arithmetic", conforming SQL shall not contain an <interval value function>.

g) Subclause 10.1, "<interval qualifier>":

i) Without Feature F052, "Intervals and datetime arithmetic", conforming SQL language shall not contain an <interval qualifier>.

34) Specifications for Feature F053, "OVERLAPS predicate":

a) Subclause 8.13, "<overlaps predicate>":

i) Without Feature F053, "OVERLAPS predicate", conforming SQL language shall not contain an <overlaps predicate>.

35) Specifications for Feature F111, "Isolation levels other than SERIALIZABLE":

a) Subclause 16.1, "<start transaction statement>":

i) Without Feature F111, "Isolation levels other than SERIALIZABLE", conforming SQL language shall not contain an <isolation level> that contains a <level of isolation> other than SERIALIZABLE.

b) Subclause 18.1, "<set session characteristics statement>":

i) Without Feature F111, "Isolation levels other than SERIALIZABLE", conforming SQL language shall not contain a <set session characteristics statement> that contains a <level of isolation> other than SERIALIZABLE.

36) Specifications for Feature F121, "Basic diagnostics management":

a) Subclause 16.1, "<start transaction statement>":

i) Without Feature F121, "Basic diagnostics management", conforming SQL language shall not contain a <diagnostics size>.

b) Subclause 22.1, "<get diagnostics statement>":

i) Without Feature F121, "Basic diagnostics management", conforming SQL language shall not contain a <get diagnostics statement>.

37) Specifications for Feature F171, "Multiple schemas per user":

a) Subclause 11.1, "<schema definition>":

i) Without Feature F171, "Multiple schemas per user", conforming SQL language shall not contain a <schema name clause> that contains a <schema name>.

38) Specifications for Feature F191, "Referential delete actions":

a) Subclause 11.8, "<referential constraint definition>":

i) Without Feature F191, "Referential delete actions", conforming SQL language shall not contain a <delete rule>.

39) Specifications for Feature F222, "INSERT statement: DEFAULT VALUES clause":

a) Subclause 14.8, "<insert statement>":

i) Without Feature F222, "INSERT statement: DEFAULT VALUES clause", conforming SQL language shall not contain a <from default>.

40) Specifications for Feature F251, "Domain support":

a) Subclause 5.4, "Names and identifiers":

i) Without Feature F251, "Domain support", conforming SQL language shall not contain a <domain name>.

b) Subclause 6.4, "<value specification> and <target specification>":

i) Without Feature F251, "Domain support", conforming SQL language shall not contain a <general value specification> that contains VALUE.

c) Subclause 11.24, "<domain definition>":

i) Without Feature F251, "Domain support", conforming SQL language shall not contain a <domain definition>.

d) Subclause 11.30, "<drop domain statement>":

i) Without Feature F251, "Domain support", conforming SQL language shall not contain a <drop domain statement>.

41) Specifications for Feature F262, "Extended CASE expression":

a) Subclause 6.11, "<case expression>":

i) Without Feature F262, "Extended CASE expression", in conforming SQL language, a <case operand> shall be a <row value predicand> that is a <row value constructor predicand> that is a single <common value expression> or <boolean predicand>.

ii) Without Feature F262, "Extended CASE expression", in conforming SQL language, a <when operand> shall be a <row value predicand> that is a <row value constructor predicand> that is a single <common value expression> or <boolean predicand>.

42) Specifications for Feature F271, "Compound character literals":

a) Subclause 5.3, "<literal>":

i) Without Feature F271, "Compound character literals", in conforming SQL language, a <character string literal> shall contain exactly one repetition of <character representation> (that is, it shall contain exactly one sequence of "<quote> <character representation>... <quote>").

43) Specifications for Feature F281, "LIKE enhancements":

a) Subclause 8.5, "<like predicate>":

i) Without Feature F281, "LIKE enhancements", conforming SQL language shall not contain a <common value expression> simply contained in the <row value predicand> immediately contained in <character like predicate> that is not a column reference.

ii) Without Feature F281, "LIKE enhancements", conforming SQL language shall not contain a <character pattern> that is not a <value specification>.

iii) Without Feature F281, "LIKE enhancements", conforming SQL language shall not contain an <escape character> that is not a <value specification>.

44) Specifications for Feature F291, "UNIQUE predicate":

a) Subclause 8.10, "<unique predicate>":

i) Without Feature F291, "UNIQUE predicate", conforming SQL language shall not contain a <unique predicate>.

NOTE 461 — The Conformance Rules of Subclause 9.10, "Grouping operations", also apply.

45) Specifications for Feature F301, "CORRESPONDING in query expressions":

a) Subclause 7.13, "<query expression>":

i) Without Feature F301, "CORRESPONDING in query expressions", conforming SQL language shall not contain a <query expression> that contains CORRESPONDING.

46) Specifications for Feature F302, "INTERSECT table operator":

a) Subclause 7.13, "<query expression>":

i) Without Feature F302, "INTERSECT table operator", conforming SQL language shall not contain a <query term> that contains INTERSECT.

47) Specifications for Feature F304, "EXCEPT ALL table operator":

a) Subclause 7.13, "<query expression>":

i) Without Feature F304, "EXCEPT ALL table operator", conforming SQL language shall not contain a <query expression> that contains EXCEPT ALL.

NOTE 462 — If DISTINCT, INTERSECT or EXCEPT is specified, then the Conformance Rules of Subclause 9.10, "Grouping operations", apply.

48) Specifications for Feature F312, "MERGE statement":

a) Subclause 14.9, "<merge statement>":

i) Without Feature F312, "MERGE statement", conforming SQL language shall not contain a <merge statement>.

49) Specifications for Feature F321, "User authorization":

a) Subclause 6.4, "<value specification> and <target specification>":

i) Without Feature F321, "User authorization", conforming SQL language shall not contain a <general value specification> that contains CURRENT_USER, SYSTEM_USER, or SESSION_USER.

NOTE 463 — Although CURRENT_USER and USER are semantically the same, without Feature F321, "User authorization", CURRENT_USER shall be specified as USER.

b) Subclause 11.5, "<default clause>":

i) Without Feature F321, "User authorization", conforming SQL language shall not contain a <default option> that contains CURRENT_USER, SESSION_USER, or SYSTEM_USER.

NOTE 464 — Although CURRENT_USER and USER are semantically the same, without Feature F321, "User authorization", CURRENT_USER shall be specified as USER.

c) Subclause 18.2, "<set session user identifier statement>":

i) Without Feature F321, "User authorization", conforming SQL language shall not contain a <set session user identifier statement>.

50) Specifications for Feature F361, "Subprogram support":

a) Subclause 20.1, "<embedded SQL host program>":

i) Without Feature F361, "Subprogram support", conforming SQL language shall not contain two <host variable definition>s that specify the same variable name.

51) Specifications for Feature F381, "Extended schema manipulation":

a) Subclause 11.2, "<drop schema statement>":

i) Without Feature F381, "Extended schema manipulation", conforming SQL language shall not contain a <drop schema statement>.

b) Subclause 11.12, "<alter column definition>":

i) Without Feature F381, "Extended schema manipulation", conforming SQL language shall not contain an <alter column definition>.

c) Subclause 11.13, "<set column default clause>":

i) Without Feature F381, "Extended schema manipulation", conforming SQL language shall not contain a <set column default clause>.

d) Subclause 11.14, "<drop column default clause>":

i) Without Feature F381, "Extended schema manipulation", conforming SQL language shall not contain a <drop column default clause>.

e) Subclause 11.15, "<add column scope clause>":

i) Without Feature F381, "Extended schema manipulation", conforming SQL language shall not contain an <add column scope clause>.

f) Subclause 11.16, "<drop column scope clause>":

i) Without Feature F381, "Extended schema manipulation", conforming SQL language shall not contain a <drop column scope clause>.

g) Subclause 11.19, "<add table constraint definition>":

i) Without Feature F381, "Extended schema manipulation", conforming SQL language shall not contain an <add table constraint definition>.

h) Subclause 11.20, "<drop table constraint definition>":

i) Without Feature F381, "Extended schema manipulation", conforming SQL language shall not contain a <drop table constraint definition>.

i) Subclause 11.51, "<alter routine statement>":

i) Without Feature F381, "Extended schema manipulation", conforming SQL language shall not contain an <alter routine statement>.

52) Specifications for Feature F391, "Long identifiers":

a) Subclause 5.2, "<token> and <separator>":

i) Without Feature F391, "Long identifiers", in a <regular identifier>, the number of <identifier part>s shall be less than 18.

ii) Without Feature F391, "Long identifiers", the <delimited identifier body> of a <delimited identifier> shall not comprise more than 18 <delimited identifier part>s.

NOTE 465 — Not every character set supported by a conforming SQL-implementation necessarily contains every character associated with <identifier start> and <identifier part> that is identified in the Syntax Rules of this Subclause. No conforming SQL-implementation shall be required to support in <identifier start> or <identifier part> any character identified in the Syntax Rules of this Subclause unless that character belongs to the character set in use for an SQL-client module or in SQL-data.

53) Specifications for Feature F392, "Unicode escapes in identifiers":

a) Subclause 5.2, "<token> and <separator>":

i) Without Feature F392, "Unicode escapes in identifiers", conforming SQL language shall not contain a <Unicode delimited identifier>.

54) Specifications for Feature F393, "Unicode escapes in literals":

a) Subclause 5.3, "<literal>":

i) Without Feature F393, "Unicode escapes in literals", conforming SQL language shall not contain a <Unicode character string literal>.

55) Specifications for Feature F401, "Extended joined table":

a) Subclause 7.7, "<joined table>":

i) Without Feature F401, "Extended joined table", conforming SQL language shall not contain a <cross join>.

ii) Without Feature F401, "Extended joined table", conforming SQL language shall not contain a <natural join>.

iii) Without Feature F401, "Extended joined table", conforming SQL language shall not contain FULL.

56) Specifications for Feature F402, "Named column joins for LOBs, arrays, and multisets":

a) Subclause 7.7, "<joined table>":

i) Without Feature F402, "Named column joins for LOBs, arrays, and multisets", conforming SQL language shall not contain a <joined table> that simply contains either <natural join> or <named columns join> in which, if *C* is a corresponding join column, the declared type of *C* is LOB-ordered, array-ordered, or multiset-ordered.

NOTE 466 — If *C* is a corresponding join column, then the Conformance Rules of Subclause 9.9, "Equality operations", also apply.

57) Specifications for Feature F411, "Time zone specification":

a) Subclause 5.3, "<literal>":

i) Without Feature F411, "Time zone specification", conforming SQL language shall not contain a <time zone interval>.

b) Subclause 6.1, "<data type>":

i) Without Feature F411, "Time zone specification", conforming SQL language shall not contain <with or without time zone>.

c) Subclause 6.27, "<numeric value function>":

i) Feature F411, "Time zone specification", conforming SQL language shall not contain an <extract expression> that specifies a <time zone field>.

d) Subclause 6.30, "<datetime value expression>":

i) Without Feature F411, "Time zone specification", conforming SQL language shall not contain a <time zone>.

e) Subclause 6.31, "<datetime value function>":

i) Without Feature F411, "Time zone specification", conforming SQL language shall not contain a <current time value function>.

ii) Without Feature F411, "Time zone specification", conforming SQL language shall not contain a <current timestamp value function>.

f) Subclause 18.4, "<set local time zone statement>":

i) Without Feature F411, "Time zone specification", conforming SQL language shall not contain a <set local time zone statement>.

58) Specifications for Feature F421, "National character":

a) Subclause 5.3, "<literal>":

i) Without Feature F421, "National character", conforming SQL language shall not contain a <national character string literal>.

b) Subclause 6.1, "<data type>":

i) Without Feature F421, "National character", conforming SQL language shall not contain a <national character string type>

c) Subclause 6.12, "<cast specification>":

i) Without Feature F421, "National character", conforming SQL language shall not contain a <cast operand> whose declared type is NATIONAL CHARACTER LARGE OBJECT.

d) Subclause 6.27, "<numeric value function>":

i) Without Feature F421, "National character", conforming SQL language shall not contain a <length expression> that simply contains a <string value expression> that has a declared type of NATIONAL CHARACTER LARGE OBJECT.

e) Subclause 8.5, "<like predicate>":

i) Without Feature F421, "National character", and Feature T042, "Extended LOB data type support", in conforming SQL language, a <character value expression> simply contained in a <like predicate> shall not be of declared type NATIONAL CHARACTER LARGE OBJECT.

59) Specifications for Feature F431, "Read-only scrollable cursors":

a) Subclause 14.1, "<declare cursor>":

i) Without Feature F431, "Read-only scrollable cursors", conforming SQL language shall not contain a <cursor scrollability>.

b) Subclause 14.3, "<fetch statement>":

i) Without Feature F431, "Read-only scrollable cursors", a <fetch statement> shall not contain a <fetch orientation>.

60) Specifications for Feature F441, "Extended set function support":

a) Subclause 7.8, "<where clause>":

i) Without Feature F441, "Extended set function support", conforming SQL language shall not contain a <value expression> directly contained in a <where clause> that contains a <column reference> that references a <derived column> that generally contains a <set function specification> without an intervening <routine invocation>.

b) Subclause 10.9, "<aggregate function>":

i) Without Feature F441, "Extended set function support", conforming SQL language shall not contain a <general set function> that contains a <computational operation> that immediately contains COUNT and does not contain a <set quantifier> that immediately contains DISTINCT.

ii) Without Feature F441, "Extended set function support", conforming SQL language shall not contain a <general set function> that does not contain a <set quantifier> that immediately contains DISTINCT and that contains a <value expression> that contains a column reference that does not reference a column of *T*.

iii) Without Feature F441, "Extended set function support", conforming SQL language shall not contain a <binary set function> that does not contain either a <dependent variable expression> or an <independent variable expression> that contains a column reference that references a column of *T*.

iv) Without Feature F441, "Extended set function support", conforming SQL language shall not contain a <value expression> simply contained in a <general set function> that contains a column reference that is an outer reference where the <value expression> is not a column reference.

v) Without Feature F441, "Extended set function support", conforming SQL language shall not contain a <numeric value expression> simply contained in a <dependent variable expression> or an <independent variable expression> that contains a column reference that is an outer reference and in which the <numeric value expression> is not a column reference.

vi) Without Feature F441, "Extended set function support", conforming SQL language shall not contain a column reference contained in an <aggregate function> that contains a reference to a column derived from a <value expression> that generally contains an <aggregate function> *SFS2* without an intervening <routine invocation>.

61) Specifications for Feature F442, "Mixed column references in set functions":

a) Subclause 10.9, "<aggregate function>":

i) Without Feature F442, "Mixed column references in set functions", conforming SQL language shall not contain a <hypothetical set function value expression list> or a <sort specification list> that simply contains a <value expression> that contains more than one column reference, one of which is an outer reference.

ii) Without Feature F442, "Mixed column references in set functions", conforming SQL language shall not contain an <inverse distribution function> that contains an <inverse distribution function argument> or a <sort specification> that contains more than one column reference, one of which is an outer reference.

iii) Without Feature F442, "Mixed column references in set functions", conforming SQL language shall not contain an <aggregate function> that contains a <general set function> whose simply contained <value expression> contains more than one column reference, one of which is an outer reference.

iv) Without Feature F442, "Mixed column references in set functions", conforming SQL language shall not contain an <aggregate function> that contains a <binary set function> whose simply contained <dependent variable expression> or <independent variable expression> contains more than one column reference, one of which is an outer reference.

62) Specifications for Feature F451, "Character set definition":

a) Subclause 11.31, "<character set definition>":

i) Without Feature F451, "Character set definition", conforming SQL language shall not contain a <character set definition>.

b) Subclause 11.32, "<drop character set statement>":

i) Without Feature F451, "Character set definition", conforming SQL language shall not contain a <drop character set statement>.

63) Specifications for Feature F461, "Named character sets":

a) Subclause 5.4, "Names and identifiers":

i) Without Feature F461, "Named character sets", conforming SQL language shall not contain a <character set name>.

b) Subclause 10.5, "<character set specification>":

i) Without Feature F461, "Named character sets", conforming SQL language shall not contain a <character set specification>.

c) Subclause 11.1, "<schema definition>":

i) Without Feature F461, "Named character sets", conforming SQL language shall not contain a <schema character set specification>.

d) Subclause 13.2, "<module name clause>":

i) Without Feature F461, "Named character sets", conforming SQL language shall not contain a <module character set specification>.

e) Subclause 18.7, "<set names statement>":

i) Without and Feature F461, "Named character sets", conforming SQL language shall not contain a <set names statement>.

f) Subclause 20.1, "<embedded SQL host program>":

i) Without Feature F461, "Named character sets", conforming SQL language shall not contain an <embedded character set declaration>.

64) Specifications for Feature F491, "Constraint management":

a) Subclause 5.4, "Names and identifiers":

i) Without Feature F491, "Constraint management", conforming SQL language shall not contain a <constraint name>.

b) Subclause 10.8, "<constraint name definition> and <constraint characteristics>":

i) Without Feature F491, "Constraint management", conforming SQL language shall not contain a <constraint name definition>.

c) Subclause 11.29, "<drop domain constraint definition>":

i) Without Feature F491, "Constraint management", conforming SQL language shall not contain a <drop domain constraint definition>.

d) Subclause 20.2, "<embedded exception declaration>":

i) Without Feature F491, "Constraint management", conforming SQL language shall not contain an <SQL condition> that contains a <constraint name>.

65) Specifications for Feature F521, "Assertions":

a) Subclause 11.37, "<assertion definition>":

i) Without Feature F521, "Assertions", conforming SQL language shall not contain an <assertion definition>.

b) Subclause 11.38, "<drop assertion statement>":

i) Without Feature F521, "Assertions", conforming SQL language shall not contain a <drop assertion statement>.

66) Specifications for Feature F531, "Temporary tables":

a) Subclause 11.3, "<table definition>":

i) Without Feature F531, "Temporary tables", conforming SQL language shall not contain a <table scope> and shall not reference any global or local temporary table.

b) Subclause 14.13, "<temporary table declaration>":

i) Without Feature F531, "Temporary tables", conforming SQL language shall not contain a <temporary table declaration>.

67) Specifications for Feature F555, "Enhanced seconds precision":

a) Subclause 5.3, "<literal>":

i) Without Feature F555, "Enhanced seconds precision", in conforming SQL language, an <unsigned integer> that is a <seconds fraction> that is contained in a <timestamp literal> shall not contain more than 6 <digit>s.

ii) Without Feature F555, "Enhanced seconds precision", in conforming SQL language, a <time literal> shall not contain a <seconds fraction>.

b) Subclause 6.1, "<data type>":

i) Without Feature F555, "Enhanced seconds precision", conforming SQL language shall not contain a <time precision> that does not specify 0 (zero).

ii) Without Feature F555, "Enhanced seconds precision", conforming SQL language shall not contain a <timestamp precision> that does not specify either 0 (zero) or 6.

c) Subclause 6.31, "<datetime value function>":

i) Without Feature F555, "Enhanced seconds precision", conforming SQL language shall not contain a <current local time value function> that contains a <time precision> that is not 0 (zero).

ii) Without Feature F555, "Enhanced seconds precision", conforming SQL language shall not contain a <current local timestamp value function> that contains a <timestamp precision> that is neither 0 (zero) nor 6.

68) Specifications for Feature F561, "Full value expressions":

a) Subclause 8.4, "<in predicate>":

i) Without Feature F561, "Full value expressions", conforming SQL language shall not contain a <row value expression> immediately contained in an <in value list> that is not a <value specification>.

NOTE 467 — Since <in predicate> is an equality operation, the Conformance Rules of Subclause 9.9, "Equality operations", also apply.

b) Subclause 10.9, "<aggregate function>":

i) Without Feature F561, "Full value expressions", or Feature F801, "Full set function", conforming SQL language shall not contain a <general set function> that immediately contains DISTINCT and contains a <value expression> that is not a column reference.

69) Specifications for Feature F571, "Truth value tests":

a) Subclause 6.34, "<boolean value expression>":

i) Without Feature F571, "Truth value tests", conforming SQL language shall not contain a <boolean test> that simply contains a <truth value>.

70) Specifications for Feature F591, "Derived tables":

a) Subclause 7.6, "<table reference>":

i) Without Feature F591, "Derived tables", conforming SQL language shall not contain a <derived table>.

71) Specifications for Feature F611, "Indicator data types":

a) Subclause 6.4, "<value specification> and <target specification>":

i) Without Feature F611, "Indicator data types", in conforming SQL language, the specific declared types of <indicator parameter>s and <indicator variable>s shall be the same implementation-defined data type.

72) Specifications for Feature F641, "Row and table constructors":

a) Subclause 7.1, "<row value constructor>":

i) Without Feature F641, "Row and table constructors", conforming SQL language shall not contain an <explicit row value constructor> that is not simply contained in a <table value constructor> and that contains more than one <row value constructor element>.

ii) Without Feature F641, "Row and table constructors", conforming SQL language shall not contain an <explicit row value constructor> that is a <row subquery>.

iii) Without Feature F641, "Row and table constructors", conforming SQL language shall not contain a <contextually typed row value constructor> that is not simply contained in a <contextually typed table value constructor> and that contains more than one <row value constructor element>.

iv) Without Feature F641, "Row and table constructors", conforming SQL language shall not contain a <contextually typed row value constructor> that is a <row subquery>.

b) Subclause 7.3, "<table value constructor>":

i) Without Feature F641, "Row and table constructors", in conforming SQL language, the <contextually typed row value expression list> of a <contextually typed table value constructor> shall contain exactly one <contextually typed row value constructor> *RVE*. *RVE* shall be of the form "(<contextually typed row value constructor element list>)".

ii) Without Feature F641, "Row and table constructors", conforming SQL language shall not contain a <table value constructor>.

73) Specifications for Feature F651, "Catalog name qualifiers":

a) Subclause 5.4, "Names and identifiers":

i) Without Feature F651, "Catalog name qualifiers", conforming SQL language shall not contain a <catalog name>.

b) Subclause 18.5, "<set catalog statement>":

i) Without Feature F651, "Catalog name qualifiers", conforming SQL language shall not contain a <set catalog statement>.

74) Specifications for Feature F661, "Simple tables":

a) Subclause 7.13, "<query expression>":

i) Without Feature F661, "Simple tables", conforming SQL language shall not contain a <simple table> that immediately contains a <table value constructor> except in an <insert statement>.

ii) Without Feature F661, "Simple tables", conforming SQL language shall not contain an <explicit table>.

75) Specifications for Feature F671, "Subqueries in CHECK constraints":

a) Subclause 11.9, "<check constraint definition>":

i) Without Feature F671, "Subqueries in CHECK constraints", conforming SQL language shall not contain a <search condition> contained in a <check constraint definition> that contains a <subquery>.

76) Specifications for Feature F672, "Retrospective check constraints":

a) Subclause 11.9, "<check constraint definition>":

i) Without Feature F672, "Retrospective check constraints", conforming SQL language shall not contain a <check constraint definition> that generally contains CURRENT_DATE, CURRENT_TIMESTAMP, or LOCALTIMESTAMP.

b) Subclause 11.37, "<assertion definition>":

i) Without Feature F672, "Retrospective check constraints", conforming SQL language shall not contain an <assertion definition> that generally contains CURRENT_DATE, CURRENT_TIMESTAMP, or LOCALTIMESTAMP.

77) Specifications for Feature F690, "Collation support":

a) Subclause 5.4, "Names and identifiers":

i) Without Feature F690, "Collation support", conforming SQL language shall not contain a <collation name>.

b) Subclause 10.7, "<collate clause>":

i) Without Feature F690, "Collation support", conforming SQL language shall not contain a <collate clause>.

c) Subclause 11.33, "<collation definition>":

i) Without Feature F690, "Collation support", conforming SQL language shall not contain a <collation definition>.

d) Subclause 11.34, "<drop collation statement>":

i) Without Feature F690, "Collation support", conforming SQL language shall not contain a <drop collation statement>.

78) Specifications for Feature F692, "Extended collation support":

a) Subclause 11.4, "<column definition>":

i) Without Feature F692, "Extended collation support", conforming SQL language shall not contain a <column definition> that immediately contains a <collate clause>.

b) Subclause 11.24, "<domain definition>":

i) Without Feature F692, "Extended collation support", conforming SQL language shall not contain a <domain definition> that immediately contains a <collate clause>.

c) Subclause 11.42, "<attribute definition>":

i) Without Feature F692, "Extended collation support", conforming SQL language shall not contain an <attribute definition> that immediately contains a <collate clause>.

79) Specifications for Feature F693, "SQL-session and client module collations":

a) Subclause 6.4, "<value specification> and <target specification>":

i) Without Feature F693, "SQL-session and client module collations", conforming SQL language shall not contain <current collation specification>.

b) Subclause 13.1, "<SQL-client module definition>":

i) Without Feature F693, "SQL-session and client module collations", conforming SQL language shall not contain a <module collation specification>.

c) Subclause 18.10, "<set session collation statement>":

i) Without Feature F693, "SQL-session and client module collations", conforming SQL language shall not contain a <set session collation statement>.

80) Specifications for Feature F695, "Translation support":

a) Subclause 5.4, "Names and identifiers":

i) Without Feature F695, "Translation support", conforming SQL language shall not contain a <transliteration name>.

ii) Without Feature F695, "Translation support", conforming SQL language shall not contain a <transcoding name>.

b) Subclause 6.29, "<string value function>":

i) Without Feature F695, "Translation support", conforming SQL language shall not contain a <character transliteration>.

ii) Without Feature F695, "Translation support", conforming SQL language shall not contain a <transcoding>.

c) Subclause 11.35, "<transliteration definition>":

i) Without Feature F695, "Translation support", conforming SQL language shall not contain a <transliteration definition>.

d) Subclause 11.36, "<drop transliteration statement>":

i) Without Feature F695, "Translation support", conforming SQL language shall not contain a <drop transliteration statement>.

81) Specifications for Feature F701, "Referential update actions":

a) Subclause 11.8, "<referential constraint definition>":

i) Without Feature F701, "Referential update actions", conforming SQL language shall not contain an <update rule>.

82) Specifications for Feature F711, "ALTER domain":

a) Subclause 11.25, "<alter domain statement>":

   i) Without Feature F711, "ALTER domain", conforming SQL language shall not contain an <alter domain statement>.

b) Subclause 11.26, "<set domain default clause>":

   i) Without Feature F711, "ALTER domain", conforming SQL language shall not contain a <set domain default clause>.

c) Subclause 11.27, "<drop domain default clause>":

   i) Without Feature F711, "ALTER domain", conforming SQL language shall not contain a <drop domain default clause>.

d) Subclause 11.28, "<add domain constraint definition>":

   i) Without Feature F711, "ALTER domain", conforming SQL language shall not contain an <add domain constraint definition>.

e) Subclause 11.29, "<drop domain constraint definition>":

   i) Without Feature F711, "ALTER domain", conforming SQL language shall not contain a <drop domain constraint definition>.

83) Specifications for Feature F721, "Deferrable constraints":

a) Subclause 10.8, "<constraint name definition> and <constraint characteristics>":

   i) Without Feature F721, "Deferrable constraints", conforming SQL language shall not contain a <constraint characteristics>.

      NOTE 468 — This means that INITIALLY IMMEDIATE NOT DEFERRABLE is implicit.

b) Subclause 16.3, "<set constraints mode statement>":

   i) Without Feature F721, "Deferrable constraints", conforming SQL language shall not contain a <set constraints mode statement>.

84) Specifications for Feature F731, "INSERT column privileges":

a) Subclause 12.3, "<privileges>":

   i) Without Feature F731, "INSERT column privileges", in conforming SQL language, an <action> that contains INSERT shall not contain a <privilege column list>.

85) Specifications for Feature F741, "Referential MATCH types":

a) Subclause 8.12, "<match predicate>":

   i) Without Feature F741, "Referential MATCH types", conforming SQL language shall not contain a <match predicate>.

      NOTE 469 — The Conformance Rules of Subclause 9.9, "Equality operations", also apply.

b) Subclause 11.8, "<referential constraint definition>":

i) Without Feature F741, "Referential MATCH types", conforming SQL language shall not contain a <references specification> that contains MATCH.

86) Specifications for Feature F751, "View CHECK enhancements":

a) Subclause 11.22, "<view definition>":

i) Without Feature F751, "View CHECK enhancements", conforming SQL language shall not contain a <levels clause>.

ii) Without Feature F751, "View CHECK enhancements", conforming SQL language shall not contain <view definition> that contains a <subquery> and contains CHECK OPTION.

87) Specifications for Feature F761, "Session management":

a) Subclause 18.1, "<set session characteristics statement>":

i) Without Feature F761, "Session management", conforming SQL language shall not contain a <set session characteristics statement>.

b) Subclause 18.5, "<set catalog statement>":

i) Without Feature F761, "Session management", conforming SQL language shall not contain a <set catalog statement>.

c) Subclause 18.6, "<set schema statement>":

i) Without Feature F761, "Session management", conforming SQL language shall not contain a <set schema statement>.

d) Subclause 18.7, "<set names statement>":

i) Without Feature F761, "Session management", conforming SQL language shall not contain a <set names statement>.

88) Specifications for Feature F771, "Connection management":

a) Subclause 5.4, "Names and identifiers":

i) Without Feature F771, "Connection management", conforming SQL language shall not contain an explicit <connection name>.

b) Subclause 17.1, "<connect statement>":

i) Without Feature F771, "Connection management", conforming SQL language shall not contain a <connect statement>.

c) Subclause 17.2, "<set connection statement>":

i) Without Feature F771, "Connection management", conforming SQL language shall not contain a <set connection statement>.

d) Subclause 17.3, "<disconnect statement>":

i) Without Feature F771, "Connection management", conforming SQL language shall not contain a <disconnect statement>.

89) Specifications for Feature F781, "Self-referencing operations":

a) Subclause 14.7, "<delete statement: searched>":

i) Without Feature F781, "Self-referencing operations", conforming SQL language shall not contain a <delete statement: searched> in which a leaf generally underlying table of *T* is an underlying table of any <query expression> generally contained in the <search condition>.

b) Subclause 14.8, "<insert statement>":

i) Without Feature F781, "Self-referencing operations", conforming SQL language shall not contain an <insert statement> in which the <table name> of a leaf generally underlying table of *T* is generally contained in the <from subquery> except as the table name of a qualifying table of a column reference.

c) Subclause 14.9, "<merge statement>":

i) Without Feature F781, "Self-referencing operations", conforming SQL language shall not contain a <merge statement> in which a leaf generally underlying table of *T* is generally contained in a <query expression> immediately contained in the <table reference> except as the <table or query name> or <correlation name> of a column reference.

ii) Without Feature F781, "Self-referencing operations", conforming SQL language shall not contain a <merge statement> in which a leaf generally underlying table of *T* is an underlying table of any <query expression> generally contained in the <search condition>.

d) Subclause 14.11, "<update statement: searched>":

i) Without Feature F781, "Self-referencing operations", conforming SQL language shall not contain an <update statement: positioned> in which a leaf generally underlying table of *T* is an underlying table of any <query expression> generally contained in the <search condition>.

e) Subclause 14.12, "<set clause list>":

i) Without Feature F781, "Self-referencing operations", conforming SQL language shall not contain a <set clause> in which a leaf generally underlying table of *T* is an underlying table of any <query expression> generally contained in any <value expression> simply contained in an <update source> or <assigned row> immediately contained in the <set clause>.

90) Specifications for Feature F791, "Insensitive cursors":

a) Subclause 14.1, "<declare cursor>":

i) Without Feature F791, "Insensitive cursors", conforming SQL language shall not contain a <cursor sensitivity> that immediately contains INSENSITIVE.

ii) Without Feature F791, "Insensitive cursors", or Feature T231, "Sensitive cursors", conforming SQL language shall not contain a <cursor sensitivity> that immediately contains ASENSITIVE.

91) Specifications for Feature F801, "Full set function":

a) Subclause 7.12, "<query specification>":

i) Without Feature F801, "Full set function", conforming SQL language shall not contain a <query specification> that contains more than 1 (one) <set quantifier> that contains DISTINCT, excluding any <subquery> of that <query specification>.

92) Specifications for Feature F821, "Local table references":

a) Subclause 5.4, "Names and identifiers":

i) Without Feature F821, "Local table references", conforming SQL language shall not contain a <local or schema qualifier> that contains a <local qualifier>.

b) Subclause 6.7, "<column reference>":

i) Without Feature F821, "Local table references", conforming SQL language shall not contain a <column reference> that simply contains MODULE.

93) Specifications for Feature F831, "Full cursor update":

a) Subclause 14.1, "<declare cursor>":

i) Without Feature F831, "Full cursor update", conforming SQL language shall not contain an <updatability clause> that contains FOR UPDATE and that contains a <cursor scrollability>.

ii) Without Feature F831, "Full cursor update", conforming SQL language shall not contain an <updatability clause> that specifies FOR UPDATE and that contains an <order by clause>.

b) Subclause 14.10, "<update statement: positioned>":

i) Without Feature F831, "Full cursor update", conforming SQL language shall not contain an <update statement: positioned> in which *CR* identifies an ordered cursor.

94) Specifications for Feature S023, "Basic structured types":

a) Subclause 5.4, "Names and identifiers":

i) Without Feature S023, "Basic structured types", conforming SQL language shall not contain a <attribute name>.

b) Subclause 6.1, "<data type>":

i) Without Feature S023, "Basic structured types", conforming SQL language shall not contain a <path-resolved user-defined type name> that identifies a structured type.

c) Subclause 6.16, "<method invocation>":

i) Without Feature S023, "Basic structured types", conforming SQL language shall not contain a <method invocation>.

d) Subclause 6.18, "<new specification>":

i) Without Feature S023, "Basic structured types", conforming SQL language shall not contain a <new specification>.

e) Subclause 10.4, "<routine invocation>":

i) Without Feature S023, "Basic structured types", conforming SQL language shall not contain a <generalized expression>.

f) Subclause 11.41, "<user-defined type definition>":

i) Without Feature S023, "Basic structured types", conforming SQL language shall not contain a <member list>.

ii) Without Feature S023, "Basic structured types", conforming SQL language shall not contain a <method specification list>.

g) Subclause 11.42, "<attribute definition>":

i) Without Feature S023, "Basic structured types", conforming SQL language shall not contain an <attribute definition>.

h) Subclause 11.50, "<SQL-invoked routine>":

i) Without Feature S023, "Basic structured types", conforming SQL language shall not contain a <method specification designator>.

i) Subclause 12.3, "<privileges>":

i) Without Feature S023, "Basic structured types", conforming SQL language shall not contain a <privileges> that contains an <action> that contains UNDER and that contains an <object name> that contains a <schema-resolved user-defined type name> that identifies a structured type.

95) Specifications for Feature S024, "Enhanced structured types":

a) Subclause 6.17, "<static method invocation>":

i) Without Feature S024, "Enhanced structured types", conforming SQL language shall not contain a <static method invocation>.

b) Subclause 9.9, "Equality operations":

i) Without Feature S024, "Enhanced structured types", in conforming SQL language, the declared type of an operand of an equality operation shall not be ST-ordered.

c) Subclause 9.10, "Grouping operations":

i) Without Feature S024, "Enhanced structured types", in conforming SQL language, the declared type of an operand of a grouping operation shall not be ST-ordered.

d) Subclause 9.11, "Multiset element grouping operations":

i) Without Feature S024, "Enhanced structured types", in conforming SQL language, the declared element type of a multiset operand of a multiset element grouping operation shall not be ST-ordered.

e) Subclause 9.12, "Ordering operations":

i) Without Feature S024, "Enhanced structured types", in conforming SQL language, the declared type of an operand of an ordering operation shall not be ST-ordered.

f) Subclause 10.6, "<specific routine designator>":

i) Without Feature S024, "Enhanced structured types", conforming SQL language shall not contain a <specific routine designator> that contains a <routine type> that immediately contains METHOD.

g) Subclause 11.41, "<user-defined type definition>":

i) Without Feature S024, "Enhanced structured types", conforming SQL language shall not contain an <instantiable clause> that contains NOT INSTANTIABLE.

ii) Without Feature S024, "Enhanced structured types", conforming SQL language shall not contain an <original method specification> that immediately contains SELF AS RESULT.

iii) Without Feature S024, "Enhanced structured types", conforming SQL language shall not contain a <method characteristics> that contains a <parameter style> that contains GENERAL.

iv) Without Feature S024, "Enhanced structured types", conforming SQL language shall not contain an <original method specification> that contains an <SQL-data access indication> that immediately contains NO SQL.

v) Without Feature S024, "Enhanced structured types", conforming SQL language shall not contain a <partial method specification> that contains INSTANCE or STATIC.

h) Subclause 11.42, "<attribute definition>":

i) Without Feature S024, "Enhanced structured types", conforming SQL language shall not contain an <attribute default>.

i) Subclause 11.43, "<alter type statement>":

i) Without Feature S024, "Enhanced structured types", conforming SQL language shall not contain an <alter type statement>.

j) Subclause 11.50, "<SQL-invoked routine>":

i) Without Feature S024, "Enhanced structured types", an <SQL parameter declaration> shall not contain RESULT.

k) Subclause 11.52, "<drop routine statement>":

i) Without Feature S024, "Enhanced structured types", conforming SQL language shall not contain a <drop routine statement> that contains a <specific routine designator> that identifies a method.

l) Subclause 12.2, "<grant privilege statement>":

i) Without Feature S024, "Enhanced structured types", conforming SQL language shall not contain a <specific routine designator> contained in a <grant privilege statement> that identifies a method.

m) Subclause 12.3, "<privileges>":

i) Without Feature S024, "Enhanced structured types", conforming SQL language shall not contain a <privileges> that contains an <action> that contains USAGE and that contains an <object name> that contains a <schema-resolved user-defined type name> that identifies a structured type.

ii) Without Feature S024, "Enhanced structured types", conforming SQL language shall not contain a <privilege method list>.

n) Subclause 14.8, "<insert statement>":

i) Without Feature S024, "Enhanced structured types", in conforming SQL language, for each column *C* identified in the explicit or implicit <insert column list>, if the declared type of *C* is

a structured type *TY*, then the declared type of the corresponding column of the <query expression> or <contextually typed table value constructor> shall be *TY*.

o) Subclause 14.9, "<merge statement>":

i) Without Feature S024, "Enhanced structured types", conforming SQL language shall not contain a <merge statement> that does not satisfy the condition: for each column *C* identified in the explicit or implicit <insert column list>, if the declared type of *C* is a structured type *TY*, then the declared type of the corresponding column of the <query expression> or <contextually typed table value constructor> is *TY*.

p) Subclause 14.12, "<set clause list>":

i) Without Feature S024, "Enhanced structured types", conforming SQL language shall not contain a <set clause> in which the declared type of the <update target> in the <set clause> is a structured type *TY* and the declared type of the <update source> or corresponding field of the <assigned row> contained in the <set clause> is not *TY*.

ii) Without Feature S024, "Enhanced structured types", conforming SQL language shall not contain a <set clause> that contains a <mutated set clause> and in which the declared type of the last <method name> identifies a structured type *TY*, and the declared type of the <update source> contained in the <set clause> is not *TY*.

96) Specifications for Feature S025, "Final structured types":

a) Subclause 11.41, "<user-defined type definition>":

i) Without Feature S025, "Final structured types", in conforming SQL language, a <user-defined type definition> that defines a structured type shall contain a <finality> that is NOT FINAL.

97) Specifications for Feature S026, "Self-referencing structured types":

a) Subclause 11.42, "<attribute definition>":

i) Without Feature S026, "Self-referencing structured types", conforming SQL language shall not contain a <data type> simply contained in an <attribute definition> that is not be a <reference type> whose <referenced type> is equivalent to the <schema-resolved user-defined type name> simply contained in the <user-defined type definition> that contains <attribute definition>.

98) Specifications for Feature S027, "Create method by specific method name":

a) Subclause 11.50, "<SQL-invoked routine>":

i) Without Feature S027, "Create method by specific method name", conforming SQL language shall not contain a <method specification designator> that contains SPECIFIC METHOD.

99) Specifications for Feature S028, "Permutable UDT options list":

a) Subclause 11.41, "<user-defined type definition>":

i) Without Feature S028, "Permutable UDT options list", conforming SQL language shall not contain a <user-defined type option list> in which <instantiable clause>, if specified, <finality>, <reference type specification>, if specified, <cast to ref>, if specified, <cast to type>, if specified, <cast to distinct>, if specified, and <cast to source>, if specified, do not appear in that sequence.

100) Specifications for Feature S041, "Basic reference types":

a) Subclause 6.1, "<data type>":

   i) Without Feature S041, "Basic reference types", conforming SQL language shall not contain a <reference type>.

b) Subclause 6.19, "<attribute or method reference>":

   i) Without Feature S041, "Basic reference types", conforming SQL language shall not contain an <attribute or method reference>.

c) Subclause 6.20, "<dereference operation>":

   i) Without Feature S041, "Basic reference types", conforming SQL language shall not contain a <dereference operation>.

d) Subclause 6.25, "<value expression>":

   i) Without Feature S041, "Basic reference types", conforming SQL language shall not contain a <reference value expression>.

e) Subclause 20.3, "<embedded SQL Ada program>":

   i) Without Feature S041, "Basic reference types", conforming SQL language shall not contain an <Ada REF variable>.

f) Subclause 20.4, "<embedded SQL C program>":

   i) Without Feature S041, "Basic reference types", conforming SQL language shall not contain a <C REF variable>.

g) Subclause 20.5, "<embedded SQL COBOL program>":

   i) Without Feature S041, "Basic reference types", conforming SQL language shall not contain a <COBOL REF variable>.

h) Subclause 20.6, "<embedded SQL Fortran program>":

   i) Without Feature S041, "Basic reference types", conforming SQL language shall not contain a <Fortran REF variable>.

i) Subclause 20.7, "<embedded SQL MUMPS program>":

   i) Without Feature S041, "Basic reference types", conforming SQL language shall not contain a <MUMPS REF variable>.

j) Subclause 20.8, "<embedded SQL Pascal program>":

   i) Without Feature S041, "Basic reference types", conforming SQL language shall not contain a <Pascal REF variable>.

k) Subclause 20.9, "<embedded SQL PL/I program>":

   i) Without Feature S041, "Basic reference types", conforming SQL language shall not contain a <PL/I REF variable>.

l0l) Specifications for Feature S043, "Enhanced reference types":

a) Subclause 6.1, "<data type>":

   i) Without Feature S043, "Enhanced reference types", conforming SQL language shall not contain a <scope clause> that is not simply contained in a <data type> that is simply contained in a <column definition>.

b) Subclause 6.12, "<cast specification>":

   i) Without Feature S043, "Enhanced reference types", in conforming SQL language, if the declared data type of <cast operand> is a reference type, then <cast target> shall contain a <data type> that is a reference type.

c) Subclause 6.21, "<method reference>":

   i) Without Feature S043, "Enhanced reference types", conforming SQL language shall not contain a <method reference>.

d) Subclause 6.22, "<reference resolution>":

   i) Without Feature S043, "Enhanced reference types", conforming SQL language shall not contain a <reference resolution>.

e) Subclause 11.3, "<table definition>":

   i) Without Feature S043, "Enhanced reference types", conforming SQL language shall not contain a <column option list> that contains a <scope clause>.

   ii) Without Feature S043, "Enhanced reference types", conforming SQL language shall not contain <reference generation> that does not contain SYSTEM GENERATED.

f) Subclause 11.15, "<add column scope clause>":

   i) Without Feature S043, "Enhanced reference types", conforming SQL language shall not contain an <add column scope clause>.

g) Subclause 11.16, "<drop column scope clause>":

   i) Without Feature S043, "Enhanced reference types", conforming SQL language shall not contain a <drop column scope clause>.

h) Subclause 11.22, "<view definition>":

   i) Without Feature S043, "Enhanced reference types", conforming SQL language shall not contain a <referenceable view specification>.

i) Subclause 11.41, "<user-defined type definition>":

   i) Without Feature S043, "Enhanced reference types", conforming SQL language shall not contain a <reference type specification>.

j) Subclause 14.8, "<insert statement>":

   i) Without Feature S043, "Enhanced reference types", conforming SQL language shall not contain an <override clause>.

102) Specifications for Feature S051, "Create table of type":

a) Subclause 11.3, "<table definition>":

   i) Without Feature S051, "Create table of type", conforming SQL language shall not contain "OF <path-resolved user-defined type name>".

103) Specifications for Feature S071, "SQL paths in function and type name resolution":

a) Subclause 6.4, "<value specification> and <target specification>":

   i) Without Feature S071, "SQL paths in function and type name resolution", conforming SQL language shall not contain a <general value specification> that contains CURRENT_PATH.

b) Subclause 10.3, "<path specification>":

   i) Without Feature S071, "SQL paths in function and type name resolution", conforming SQL language shall not contain a <path specification>.

c) Subclause 11.1, "<schema definition>":

   i) Without Feature S071, "SQL paths in function and type name resolution", conforming SQL language shall not contain a <schema path specification>.

d) Subclause 11.5, "<default clause>":

   i) Without Feature S071, "SQL paths in function and type name resolution", conforming SQL language shall not contain a <default option> that contains CURRENT_PATH.

e) Subclause 13.1, "<SQL-client module definition>":

   i) Without Feature S071, "SQL paths in function and type name resolution", conforming SQL language shall not contain a <module path specification>.

f) Subclause 18.8, "<set path statement>":

   i) Without Feature S071, "SQL paths in function and type name resolution", Conforming SQL language shall not contain a <set path statement>.

g) Subclause 20.1, "<embedded SQL host program>":

   i) Without Feature S071, "SQL paths in function and type name resolution", conforming SQL language shall not contain an <embedded path specification>.

104) Specifications for Feature S081, "Subtables":

a) Subclause 11.3, "<table definition>":

   i) Without Feature S081, "Subtables", conforming SQL language shall not contain a <subtable clause>.

b) Subclause 12.2, "<grant privilege statement>":

   i) Without Feature S081, "Subtables", conforming SQL language shall not contain a <grant privilege statement> that contains WITH HIERARCHY OPTION.

c) Subclause 12.3, "<privileges>":

i) Without Feature S081, "Subtables", conforming SQL language shall not contain a <privileges> that contains an <action> that contains UNDER and that contains an <object name> that contains a <table name>.

d) Subclause 12.7, "<revoke statement>":

i) Without Feature S081, "Subtables", conforming SQL language shall not contain a <revoke option extension> that contains HIERARCHY OPTION FOR.

105) Specifications for Feature S091, "Basic array support":

a) Subclause 6.1, "<data type>":

i) Without Feature S091, "Basic array support", conforming SQL language shall not contain an <array type>.

b) Subclause 6.5, "<contextually typed value specification>":

i) Without Feature S091, "Basic array support", conforming SQL language shall not contain an <empty specification> that simply contains ARRAY.

c) Subclause 6.23, "<array element reference>":

i) Without Feature S091, "Basic array support", conforming SQL language shall not contain an <array element reference>.

d) Subclause 6.27, "<numeric value function>":

i) Without Feature S091, "Basic array support", or Feature S271, "Basic multiset support", conforming SQL language shall not contain a <cardinality expression>.

e) Subclause 6.35, "<array value expression>":

i) Without Feature S091, "Basic array support", conforming SQL language shall not contain an <array value expression>.

f) Subclause 6.36, "<array value constructor>":

i) Without Feature S091, "Basic array support", conforming SQL language shall not contain an <array value constructor by enumeration>.

g) Subclause 7.6, "<table reference>":

i) Without Feature S091, "Basic array support", or Feature S271, "Basic multiset support", conforming SQL language shall not contain a <collection derived table>.

h) Subclause 14.12, "<set clause list>":

i) Without Feature S091, "Basic array support", conforming SQL language shall not contain an <update target> that immediately contains a <simple value specification>.

106) Specifications for Feature S092, "Arrays of user-defined types":

a) Subclause 6.1, "<data type>":

i) Without Feature S092, "Arrays of user-defined types", conforming SQL language shall not contain an <array type> that is based on a <data type> that contains a <path-resolved user-defined type name>.

107) Specifications for Feature S094, "Arrays of reference types":

a) Subclause 6.1, "<data type>":

i) Without Feature S094, "Arrays of reference types", conforming SQL language shall not contain an <array type> that is based on a <data type> that contains a <reference type>.

108) Specifications for Feature S095, "Array constructors by query":

a) Subclause 6.36, "<array value constructor>":

i) Without Feature S095, "Array constructors by query", conforming SQL language shall not contain an <array value constructor by query>.

109) Specifications for Feature S096, "Optional array bounds":

a) Subclause 6.1, "<data type>":

i) Without Feature S096, "Optional array bounds", conforming SQL language shall not contain an <array type> that does not immediately contain <maximum cardinality>.

110) Specifications for Feature S097, "Array element assignment":

a) Subclause 6.4, "<value specification> and <target specification>":

i) Without Feature S097, "Array element assignment", conforming SQL language shall not contain a <target array element specification>.

111) Specifications for Feature S111, "ONLY in query expressions":

a) Subclause 7.6, "<table reference>":

i) Without Feature S111, "ONLY in query expressions", conforming SQL language shall not contain a <table reference> that contains an <only spec>.

b) Subclause 14.6, "<delete statement: positioned>":

i) Without Feature S111, "ONLY in query expressions", conforming SQL language shall not contain a <target table> that contains ONLY.

112) Specifications for Feature S151, "Type predicate":

a) Subclause 8.18, "<type predicate>":

i) Without Feature S151, "Type predicate", conforming SQL language shall not contain a <type predicate>.

113) Specifications for Feature S161, "Subtype treatment":

a) Subclause 6.15, "<subtype treatment>":

i) Without Feature S161, "Subtype treatment", conforming SQL Language shall not contain a <subtype treatment>.

114) Specifications for Feature S162, "Subtype treatment for references":

a) Subclause 6.15, "<subtype treatment>":

i) Without Feature S162, "Subtype treatment for references", conforming SQL language shall not contain a <target subtype> that contains a <reference type>.

115) Specifications for Feature S201, "SQL-invoked routines on arrays":

a) Subclause 10.4, "<routine invocation>":

i) Without Feature S201, "SQL-invoked routines on arrays", conforming SQL language shall not contain an <SQL argument> whose declared type is an array type.

b) Subclause 11.50, "<SQL-invoked routine>":

i) Without Feature S201, "SQL-invoked routines on arrays", conforming SQL language shall not contain a <parameter type> that is based on an array type.

ii) Without Feature S201, "SQL-invoked routines on arrays", conforming SQL language shall not contain a <returns data type> that is based on an array type.

116) Specifications for Feature S202, "SQL-invoked routines on multisets":

a) Subclause 10.4, "<routine invocation>":

i) Without Feature S202, "SQL-invoked routines on multisets", conforming SQL language shall not contain an <SQL argument> whose declared type is a multiset type.

b) Subclause 11.50, "<SQL-invoked routine>":

i) Without Feature S202, "SQL-invoked routines on multisets", conforming SQL language shall not contain a <parameter type> that is based on a multiset type.

ii) Without Feature S202, "SQL-invoked routines on multisets", conforming SQL language shall not contain a <returns data type> that is based on a multiset type.

117) Specifications for Feature S211, "User-defined cast functions":

a) Subclause 11.53, "<user-defined cast definition>":

i) Without Feature S211, "User-defined cast functions", conforming SQL language shall not contain a <user-defined cast definition>.

b) Subclause 11.54, "<drop user-defined cast statement>":

i) Without Feature S211, "User-defined cast functions", conforming SQL language shall not contain a <drop user-defined cast statement>.

118) Specifications for Feature S231, "Structured type locators":

a) Subclause 11.50, "<SQL-invoked routine>":

i) Without Feature S231, "Structured type locators", conforming SQL language shall not contain a <parameter type> that contains a <locator indication> and that simply contains a <data type> that identifies a structured type.

ii) Without Feature S231, "Structured type locators", conforming SQL language shall not contain a <returns data type> that contains a <locator indication> and that simply contains a <data type> that identifies a structured type.

b) Subclause 13.3, "<externally-invoked procedure>":

i) Without Feature S231, "Structured type locators", conforming SQL language shall not contain a <host parameter data type> that simply contains a <data type> that specifies a structured type and that contains a <locator indication>.

c) Subclause 20.3, "<embedded SQL Ada program>":

i) Without Feature S231, "Structured type locators", conforming SQL language shall not contain a <path-resolved user-defined type name> simply contained in an <Ada user-defined type locator variable> that identifies a structured type.

d) Subclause 20.4, "<embedded SQL C program>":

i) Without Feature S231, "Structured type locators", conforming SQL language shall not contain a <path-resolved user-defined type name> simply contained in a <C user-defined type locator variable> that identifies a structured type.

e) Subclause 20.5, "<embedded SQL COBOL program>":

i) Without Feature S231, "Structured type locators", conforming SQL language shall not contain a <path-resolved user-defined type name> simply contained in a <COBOL user-defined type locator variable> that identifies a structured type.

f) Subclause 20.6, "<embedded SQL Fortran program>":

i) Without Feature S231, "Structured type locators", conforming SQL language shall not contain a <path-resolved user-defined type name> simply contained in a <Fortran user-defined type locator variable> that identifies a structured type.

g) Subclause 20.7, "<embedded SQL MUMPS program>":

i) Without Feature S231, "Structured type locators", conforming SQL language shall not contain a <path-resolved user-defined type name> simply contained in a <MUMPS user-defined type locator variable> that identifies a structured type.

h) Subclause 20.8, "<embedded SQL Pascal program>":

i) Without Feature S231, "Structured type locators", conforming SQL language shall not contain a <path-resolved user-defined type name> simply contained in a <Pascal user-defined type locator variable> that identifies a structured type.

i) Subclause 20.9, "<embedded SQL PL/I program>":

i) Without Feature S231, "Structured type locators", conforming SQL language shall not contain a <path-resolved user-defined type name> simply contained in a <PL/I user-defined type locator variable> that identifies a structured type.

119) Specifications for Feature S232, "Array locators":

a) Subclause 11.50, "<SQL-invoked routine>":

i) Without Feature S232, "Array locators", conforming SQL language shall not contain a <parameter type> that contains a <locator indication> and that simply contains a <data type> that identifies an array type.

ii) Without Feature S232, "Array locators", conforming SQL language shall not contain a <returns data type> that contains a <locator indication> and that simply contains a <data type> that identifies an array type.

b) Subclause 13.3, "<externally-invoked procedure>":

i) Without Feature S232, "Array locators", conforming SQL language shall not contain a <host parameter data type> that simply contains an <array type> and that contains a <locator indication>.

c) Subclause 20.3, "<embedded SQL Ada program>":

i) Without Feature S232, "Array locators", conforming SQL language shall not contain an <Ada array locator variable>.

d) Subclause 20.4, "<embedded SQL C program>":

i) Without Feature S232, "Array locators", conforming SQL language shall not contain an <C array locator variable>.

e) Subclause 20.5, "<embedded SQL COBOL program>":

i) Without Feature S232, "Array locators", conforming SQL language shall not contain a <COBOL array locator variable>.

f) Subclause 20.6, "<embedded SQL Fortran program>":

i) Without Feature S232, "Array locators", conforming SQL language shall not contain a <Fortran array locator variable>.

g) Subclause 20.7, "<embedded SQL MUMPS program>":

i) Without Feature S232, "Array locators", conforming SQL language shall not contain a <MUMPS array locator variable>.

h) Subclause 20.8, "<embedded SQL Pascal program>":

i) Without Feature S232, "Array locators", conforming SQL language shall not contain a <Pascal array locator variable>.

i) Subclause 20.9, "<embedded SQL PL/I program>":

i) Without Feature S232, "Array locators", conforming SQL language shall not contain a <PL/I array locator variable>.

120) Specifications for Feature S233, "Multiset locators":

a) Subclause 11.50, "<SQL-invoked routine>":

i) Without Feature S233, "Multiset locators", conforming SQL language shall not contain a <parameter type> that contains a <locator indication> and that simply contains a <data type> that identifies a multiset type.

ii) Without Feature S233, "Multiset locators", conforming SQL language shall not contain a <returns data type> that contains a <locator indication> and that simply contains a <data type> that identifies a multiset type.

b) Subclause 13.3, "<externally-invoked procedure>":

i) Without Feature S233, "Multiset locators", conforming SQL language shall not contain a <host parameter data type> that simply contains a <multiset type> and that contains a <locator indication>.

c) Subclause 20.3, "<embedded SQL Ada program>":

i) Without Feature S233, "Multiset locators", conforming SQL language shall not contain an <Ada multiset locator variable>.

d) Subclause 20.4, "<embedded SQL C program>":

i) Without Feature S233, "Multiset locators", conforming SQL language shall not contain a <C multiset locator variable>.

e) Subclause 20.5, "<embedded SQL COBOL program>":

i) Without Feature S233, "Multiset locators", conforming SQL language shall not contain a <COBOL multiset locator variable>.

f) Subclause 20.6, "<embedded SQL Fortran program>":

i) Without Feature S233, "Multiset locators", conforming SQL language shall not contain a <Fortran multiset locator variable>.

g) Subclause 20.7, "<embedded SQL MUMPS program>":

i) Without Feature S233, "Multiset locators", conforming SQL language shall not contain a <MUMPS multiset locator variable>.

h) Subclause 20.8, "<embedded SQL Pascal program>":

i) Without Feature S233, "Multiset locators", conforming SQL language shall not contain a <Pascal multiset locator variable>.

i) Subclause 20.9, "<embedded SQL PL/I program>":

i) Without Feature S233, "Multiset locators", conforming SQL language shall not contain a <PL/I multiset locator variable>.

121) Specifications for Feature S241, "Transform functions":

a) Subclause 6.4, "<value specification> and <target specification>":

i) Without Feature S241, "Transform functions", conforming SQL language shall not contain CURRENT_DEFAULT_TRANSFORM_GROUP.

ii) Without Feature S241, "Transform functions", conforming SQL language shall not contain CURRENT_TRANSFORM_GROUP_FOR_TYPE.

b) Subclause 11.50, "<SQL-invoked routine>":

i) Without Feature S241, "Transform functions", conforming SQL language shall not contain a <transform group specification>.

c) Subclause 11.57, "<transform definition>":

i) Without Feature S241, "Transform functions", conforming SQL language shall not contain a <transform definition>.

d) Subclause 11.61, "<drop transform statement>":

i) Without Feature S241, "Transform functions", conforming SQL language shall not contain a <drop transform statement>.

e) Subclause 13.1, "<SQL-client module definition>":

i) Without Feature S241, "Transform functions", conforming SQL language shall not contain a <module transform group specification>.

f) Subclause 18.9, "<set transform group statement>":

i) Without Feature S241, "Transform functions", conforming SQL language shall not contain a <set transform group statement>.

g) Subclause 20.1, "<embedded SQL host program>":

i) Without Feature S241, "Transform functions", conforming SQL language shall not contain a <embedded transform group specification>.

h) Subclause 20.3, "<embedded SQL Ada program>":

i) Without Feature S241, "Transform functions", conforming SQL language shall not contain an <Ada user-defined type variable>.

i) Subclause 20.4, "<embedded SQL C program>":

i) Without Feature S241, "Transform functions", conforming SQL language shall not contain a <C user-defined type variable>.

j) Subclause 20.5, "<embedded SQL COBOL program>":

i) Without Feature S241, "Transform functions", conforming SQL language shall not contain a <COBOL user-defined type variable>.

k) Subclause 20.6, "<embedded SQL Fortran program>":

i) Without Feature S241, "Transform functions", conforming SQL language shall not contain a <Fortran user-defined type variable>.

l) Subclause 20.7, "<embedded SQL MUMPS program>":

i) Without Feature S241, "Transform functions", conforming SQL language shall not contain a <MUMPS user-defined type variable>.

m) Subclause 20.8, "<embedded SQL Pascal program>":

i) Without Feature S241, "Transform functions", conforming SQL language shall not contain a <Pascal user-defined type variable>.

n) Subclause 20.9, "<embedded SQL PL/I program>":

i) Without Feature S241, "Transform functions", conforming SQL language shall not contain a <PL/I user-defined type variable>.

122) Specifications for Feature S242, "Alter transform statement":

a) Subclause 11.58, "<alter transform statement>":

i) Without Feature S242, "Alter transform statement", conforming SQL language shall not contain an <alter transform statement>.

123) Specifications for Feature S251, "User-defined orderings":

a) Subclause 11.55, "<user-defined ordering definition>":

i) Without Feature S251, "User-defined orderings", conforming SQL shall not contain a <user-defined ordering definition>.

NOTE 470 — If MAP is specified, then the Conformance Rules of Subclause 9.9, "Equality operations", apply. If ORDER FULL BY MAP is specified, then the Conformance Rules of Subclause 9.12, "Ordering operations", also apply.

b) Subclause 11.56, "<drop user-defined ordering statement>":

i) Without Feature S251, "User-defined orderings", conforming SQL language shall not contain a <drop user-defined ordering statement>.

124) Specifications for Feature S261, "Specific type method":

a) Subclause 6.29, "<string value function>":

i) Without Feature S261, "Specific type method", conforming SQL language shall not contain a <specific type method>.

125) Specifications for Feature S271, "Basic multiset support":

a) Subclause 6.1, "<data type>":

i) Without Feature S271, "Basic multiset support", conforming SQL language shall not contain a <multiset type>.

b) Subclause 6.5, "<contextually typed value specification>":

i) Without Feature S271, "Basic multiset support", conforming SQL language shall not contain an <empty specification> that simply contains MULTISET.

c) Subclause 6.24, "<multiset element reference>":

i) Without Feature S271, "Basic multiset support", conforming SQL language shall not contain a <multiset element reference>.

d) Subclause 6.27, "<numeric value function>":

i) Without Feature S091, "Basic array support", or Feature S271, "Basic multiset support", conforming SQL language shall not contain a <cardinality expression>.

e) Subclause 6.38, "<multiset value function>":

i) Without Feature S271, "Basic multiset support", conforming SQL language shall not contain a <multiset value function>.

NOTE 471 — The Conformance Rules of Subclause 9.11, "Multiset element grouping operations", also apply.

f) Subclause 6.39, "<multiset value constructor>":

i) Without Feature S271, "Basic multiset support", conforming SQL language shall not contain a <multiset value constructor>.

g) Subclause 7.6, "<table reference>":

i) Without Feature S091, "Basic array support", or Feature S271, "Basic multiset support", conforming SQL language shall not contain a <collection derived table>.

h) Subclause 8.15, "<member predicate>":

i) Without Feature S271, "Basic multiset support", conforming SQL language shall not contain a <member predicate>.

NOTE 472 — The Conformance Rules of Subclause 9.9, "Equality operations", also apply.

i) Subclause 8.17, "<set predicate>":

i) Without Feature S271, "Basic multiset support", conforming SQL language shall not contain a <set predicate>.

NOTE 473 — The Conformance Rules of Subclause 9.11, "Multiset element grouping operations", also apply.

j) Subclause 10.9, "<aggregate function>":

i) Without Feature S271, "Basic multiset support", conforming SQL language shall not contain a <computational operation> that immediately contains COLLECT.

126) Specifications for Feature S272, "Multisets of user-defined types":

a) Subclause 6.1, "<data type>":

i) Without Feature S272, "Multisets of user-defined types", conforming SQL language shall not contain a <multiset type> that is based on a <data type> that contains a <path-resolved user-defined type name>.

127) Specifications for Feature S274, "Multisets of reference types":

a) Subclause 6.1, "<data type>":

i) Without Feature S274, "Multisets of reference types", conforming SQL language shall not contain a <multiset type> that is based on a <data type> that contains a <reference type>.

128) Specifications for Feature S275, "Advanced multiset support":

a) Subclause 6.37, "<multiset value expression>":

i) Without Feature S275, "Advanced multiset support", conforming SQL language shall not contain MULTISET UNION, MULTISET INTERSECTION, or MULTISET EXCEPT.

NOTE 474 — If MULTISET UNION DISTINCT, MULTISET INTERSECTION, or MULTISET EXCEPT is specified, then the Conformance Rules of Subclause 9.11, "Multiset element grouping operations", also apply.

b) Subclause 8.16, "<submultiset predicate>":

i) Without Feature S275, "Advanced multiset support", conforming SQL language shall not contain a <submultiset predicate>.

NOTE 475 — The Conformance Rules of Subclause 9.11, "Multiset element grouping operations", also apply.

c) Subclause 9.9, "Equality operations":

i) Without Feature S275, "Advanced multiset support", in conforming SQL language, the declared type of an operand of an equality operation shall not be multiset-ordered.

NOTE 476 — If the declared type of an operand *OP* of an equality operation is a multiset type, then *OP* is a multiset operand of a multiset element grouping operation. The Conformance Rules of Subclause 9.11, "Multiset element grouping operations", apply.

d) Subclause 10.9, "<aggregate function>":

i) Without Feature S275, "Advanced multiset support", conforming SQL language shall not contain a <computational operation> that immediately contains FUSION or INTERSECTION.

NOTE 477 — If INTERSECTION is specified, then the Conformance Rules of Subclause 9.11, "Multiset element grouping operations", also apply.

129) Specifications for Feature S281, "Nested collection types":

a) Subclause 6.1, "<data type>":

i) Without Feature S281, "Nested collection types", conforming SQL language shall not contain a collection type that is based on a <data type> that contains a <collection type>.

130) Specifications for Feature S291, "Unique constraint on entire row":

a) Subclause 11.7, "<unique constraint definition>":

i) Without Feature S291, "Unique constraint on entire row", conforming SQL language shall not contain UNIQUE(VALUE).

131) Specifications for Feature T031, "BOOLEAN data type":

a) Subclause 5.3, "<literal>":

i) Without Feature T031, "BOOLEAN data type", conforming SQL language shall not contain a <boolean literal>.

b) Subclause 6.1, "<data type>":

i) Without Feature T031, "BOOLEAN data type", conforming SQL language shall not contain a <boolean type>.

c) Subclause 6.25, "<value expression>":

i) Without Feature T031, "BOOLEAN data type", conforming SQL language shall not contain a <value expression> that is a <boolean value expression>.

d) Subclause 6.34, "<boolean value expression>":

i) Without Feature T031, "BOOLEAN data type", conforming SQL language shall not contain a <boolean primary> that simply contains a <nonparenthesized value expression primary>.

e) Subclause 7.1, "<row value constructor>":

i) Without Feature T031, "BOOLEAN data type", conforming SQL language shall not contain a <row value constructor predicand> that immediately contains a <boolean predicand>.

f) Subclause 10.9, "<aggregate function>":

i) Without Feature T031, "BOOLEAN data type", conforming SQL language shall not contain a <computational operation> that immediately contains EVERY, ANY, or SOME.

132) Specifications for Feature T041, "Basic LOB data type support":

a) Subclause 5.3, "<literal>":

i) Without Feature T041, "Basic LOB data type support", conforming SQL language shall not contain a <binary string literal>.

b) Subclause 6.1, "<data type>":

i) Without Feature T041, "Basic LOB data type support", conforming SQL language shall not contain a <binary large object string type>, a <character large object type>, or a <national character large object type>.

c) Subclause 11.50, "<SQL-invoked routine>":

i) Without Feature T041, "Basic LOB data type support", conforming SQL language shall not contain a <parameter type> that contains a <locator indication> and that simply contains a <data type> that identifies a large object type.

ii) Without Feature T041, "Basic LOB data type support", conforming SQL language shall not contain a <returns data type> that contains a <locator indication> and that simply contains a <data type> that identifies a large object type.

d) Subclause 20.3, "<embedded SQL Ada program>":

i) Without Feature T041, "Basic LOB data type support", conforming SQL language shall not contain an <Ada BLOB variable>.

ii) Without Feature T041, "Basic LOB data type support", conforming SQL language shall not contain an <Ada CLOB variable>.

iii) Without Feature T041, "Basic LOB data type support", conforming SQL language shall not contain an <Ada BLOB locator variable>.

iv) Without Feature T041, "Basic LOB data type support", conforming SQL language shall not contain an <Ada CLOB locator variable>.

e) Subclause 20.4, "<embedded SQL C program>":

i) Without Feature T041, "Basic LOB data type support", conforming SQL language shall not contain a <C BLOB variable>.

ii) Without Feature T041, "Basic LOB data type support", conforming SQL language shall not contain a <C CLOB variable>.

iii) Without Feature T041, "Basic LOB data type support", conforming SQL language shall not contain a <C BLOB locator variable>.

iv) Without Feature T041, "Basic LOB data type support", conforming SQL language shall not contain a <C CLOB locator variable>.

f) Subclause 20.5, "<embedded SQL COBOL program>":

i) Without Feature T041, "Basic LOB data type support", conforming SQL language shall not contain a <COBOL BLOB variable>.

ii) Without Feature T041, "Basic LOB data type support", conforming SQL language shall not contain a <COBOL CLOB variable>.

iii) Without Feature T041, "Basic LOB data type support", conforming SQL language shall not contain a <COBOL BLOB locator variable>.

iv) Without Feature T041, "Basic LOB data type support", conforming SQL language shall not contain a <COBOL CLOB locator variable>.

g) Subclause 20.6, "<embedded SQL Fortran program>":

i) Without Feature T041, "Basic LOB data type support", conforming SQL language shall not contain a <Fortran BLOB variable>.

ii) Without Feature T041, "Basic LOB data type support", conforming SQL language shall not contain a <Fortran CLOB variable>.

iii) Without Feature T041, "Basic LOB data type support", conforming SQL language shall not contain a <Fortran BLOB locator variable>.

iv) Without Feature T041, "Basic LOB data type support", conforming SQL language shall not contain a <Fortran CLOB locator variable>.

h) Subclause 20.7, "<embedded SQL MUMPS program>":

i) Without Feature T041, "Basic LOB data type support", conforming SQL language shall not contain a <MUMPS BLOB variable>.

ii) Without Feature T041, "Basic LOB data type support", conforming SQL language shall not contain a <MUMPS CLOB variable>.

iii) Without Feature T041, "Basic LOB data type support", conforming SQL language shall not contain a <MUMPS BLOB locator variable>.

iv) Without Feature T041, "Basic LOB data type support", conforming SQL language shall not contain a and <MUMPS CLOB locator variable>.

i) Subclause 20.8, "<embedded SQL Pascal program>":

i) Without Feature T041, "Basic LOB data type support", conforming SQL language shall not contain a <Pascal BLOB variable>.

ii) Without Feature T041, "Basic LOB data type support", conforming SQL language shall not contain a <Pascal CLOB variable>.

iii) Without Feature T041, "Basic LOB data type support", conforming SQL language shall not contain a <Pascal BLOB locator variable>.

iv) Without Feature T041, "Basic LOB data type support", conforming SQL language shall not contain a <Pascal BLOB variable>, <Pascal CLOB variable>, <Pascal CLOB locator variable>.

j) Subclause 20.9, "<embedded SQL PL/I program>":

i) Without Feature T041, "Basic LOB data type support", conforming SQL language shall not contain a <PL/I BLOB variable>.

ii) Without Feature T041, "Basic LOB data type support", conforming SQL language shall not contain a <PL/I CLOB variable>.

iii) Without Feature T041, "Basic LOB data type support", conforming SQL language shall not contain a <PL/I BLOB locator variable>.

iv) Without Feature T041, "Basic LOB data type support", conforming SQL language shall not contain a <PL/I CLOB locator variable>.

133) Specifications for Feature T042, "Extended LOB data type support":

a) Subclause 6.12, "<cast specification>":

i) Without Feature T042, "Extended LOB data type support", conforming SQL language shall not contain a <cast operand> whose declared type is BINARY LARGE OBJECT or CHARACTER LARGE OBJECT.

ii) Without Feature T042, "Extended LOB data type support", conforming SQL language shall not contain a <cast operand> whose declared type is NATIONAL CHARACTER LARGE OBJECT.

b) Subclause 6.29, "<string value function>":

i) Without Feature T042, "Extended LOB data type support", conforming SQL language shall not contain a <blob value function>.

c) Subclause 8.5, "<like predicate>":

i) Without Feature T042, "Extended LOB data type support", conforming SQL language shall not contain an <octet like predicate>.

ii) Without Feature T042, "Extended LOB data type support", in conforming SQL language, a <character value expression> simply contained in a <like predicate> shall not be of declared type CHARACTER LARGE OBJECT

iii) Without Feature F421, "National character", and Feature T042, "Extended LOB data type support", in conforming SQL language, a <character value expression> simply contained in a <like predicate> shall not be of declared type NATIONAL CHARACTER LARGE OBJECT.

d) Subclause 8.6, "<similar predicate>":

i) Without Feature T042, "Extended LOB data type support", in conforming SQL language, a <character value expression> simply contained in a <similar predicate> shall not be of declared type CHARACTER LARGE OBJECT.

e) Subclause 9.9, "Equality operations":

i) Without Feature T042, "Extended LOB data type support", in conforming SQL language, the declared type of an operand of an equality operation shall not be LOB-ordered.

134) Specifications for Feature T051, "Row types":

a) Subclause 5.4, "Names and identifiers":

i) Without Feature T051, "Row types", conforming SQL language shall not contain a <field name>.

b) Subclause 6.1, "<data type>":

i) Without Feature T051, "Row types", conforming SQL language shall not contain a <row type>.

c) Subclause 6.2, "<field definition>":

i) Without Feature T051, "Row types", conforming SQL language shall not contain a <field definition>.

d) Subclause 6.14, "<field reference>":

i) Without Feature T051, "Row types", conforming SQL language shall not contain a <field reference>.

e) Subclause 7.1, "<row value constructor>":

i) Without Feature T051, "Row types", conforming SQL language shall not contain an <explicit row value constructor> that immediately contains ROW.

ii) Without Feature T051, "Row types", conforming SQL language shall not contain a <contextually typed row value constructor> that immediately contains ROW.

f) Subclause 7.2, "<row value expression>":

i) Without Feature T051, "Row types", conforming SQL language shall not contain a <row value special case>.

g) Subclause 7.12, "<query specification>":

i) Without Feature T051, "Row types", conforming SQL language shall not contain an <all fields reference>.

135) Specifications for Feature T052, "MAX and MIN for row types":

a) Subclause 10.9, "<aggregate function>":

i) Without Feature T052, "MAX and MIN for row types", conforming SQL language shall not contain a <computational operation> that immediately contains MAX or MIN in which the declared type of the <value expression> is a row type.

NOTE 478 — If DISTINCT is specified, then the Conformance Rules of Subclause 9.10, "Grouping operations", also apply. If MAX or MIN is specified, then the Conformance Rules of Subclause 9.12, "Ordering operations", also apply.

136) Specifications for Feature T053, "Explicit aliases for all-fields reference":

a) Subclause 7.12, "<query specification>":

i) Without Feature T053, "Explicit aliases for all-fields reference", conforming SQL language shall not contain an <all fields column name list>.

137) Specifications for Feature T061, "UCS support":

a) Subclause 6.1, "<data type>":

i) Without Feature T061, "UCS support", conforming SQL language shall not contain a <char length units>.

b) Subclause 6.29, "<string value function>":

i) Without Feature T061, "UCS support", conforming SQL language shall not contain a <normalize function>.

c) Subclause 8.11, "<normalized predicate>":

i) Without Feature T061, "UCS support", conforming SQL language shall not contain a <normalized predicate>.

138) Specifications for Feature T071, "BIGINT data type":

a) Subclause 6.1, "<data type>":

i) Without Feature T071, "BIGINT data type", conforming SQL language shall not contain BIGINT.

b) Subclause 20.3, "<embedded SQL Ada program>":

i) Without Feature T071, "BIGINT data type", conforming SQL language shall not contain an <Ada qualified type specification> that contains Interfaces.SQL.BIGINT.

ii) Without Feature T071, "BIGINT data type", conforming SQL language shall not contain an <Ada unqualified type specification> that contains BIGINT.

c) Subclause 20.4, "<embedded SQL C program>":

i) Without Feature T071, "BIGINT data type", conforming SQL language shall not contain a <C numeric variable> that contains `long long`.

139) Specifications for Feature T111, "Updatable joins, unions, and columns":

a) Subclause 7.12, "<query specification>":

i) Without Feature T111, "Updatable joins, unions, and columns", in conforming SQL language, a <query specification> *QS* is not updatable if it is not simply updatable.

NOTE 479 — If a <set quantifier> DISTINCT is specified, then the Conformance Rules of Subclause 9.10, "Grouping operations", also apply.

b) Subclause 7.13, "<query expression>":

i) Without Feature T111, "Updatable joins, unions, and columns", in conforming SQL language, a <query expression body> that immediately contains UNION is not updatable.

140) Specifications for Feature T121, "WITH (excluding RECURSIVE) in query expression":

a) Subclause 5.4, "Names and identifiers":

i) Without Feature T121, "WITH (excluding RECURSIVE) in query expression", conforming SQL language shall not contain a <query name>.

b) Subclause 7.6, "<table reference>":

i) Without Feature T121, "WITH (excluding RECURSIVE) in query expression", conforming SQL language shall not contain a <query name>.

c) Subclause 7.13, "<query expression>":

i) Without Feature T121, "WITH (excluding RECURSIVE) in query expression", in conforming SQL language, a <query expression> shall not contain a <with clause>.

141) Specifications for Feature T122, "WITH (excluding RECURSIVE) in subquery":

a) Subclause 7.13, "<query expression>":

i) Without Feature T122, "WITH (excluding RECURSIVE) in subquery", in conforming SQL language, a <query expression> contained in a <subquery>, a <multiset value constructor by query>, or an <array value constructor by query> shall not contain a <with clause>.

142) Specifications for Feature T131, "Recursive query":

a) Subclause 7.13, "<query expression>":

i) Without Feature T131, "Recursive query", conforming SQL language shall not contain a <query expression> that contains RECURSIVE.

b) Subclause 11.22, "<view definition>":

i) Without Feature T131, "Recursive query", conforming SQL language shall not contain a <view definition> that immediately contains RECURSIVE.

143) Specifications for Feature T132, "Recursive query in subquery":

a) Subclause 7.13, "<query expression>":

i) Without Feature T132, "Recursive query in subquery", in conforming SQL language, a <query expression> contained in a <subquery>, a <multiset value constructor by query>, or an <array value constructor by query> shall not contain RECURSIVE.

144) Specifications for Feature T141, "SIMILAR predicate":

a) Subclause 8.6, "<similar predicate>":

i) Without Feature T141, "SIMILAR predicate", conforming SQL language shall not contain a <similar predicate>.

145) Specifications for Feature T151, "DISTINCT predicate":

a) Subclause 8.14, "<distinct predicate>":

i) Without Feature T151, "DISTINCT predicate", conforming SQL language shall not contain a <distinct predicate>.

NOTE 480 — The Conformance Rules of Subclause 9.9, "Equality operations", also apply.

146) Specifications for Feature T152, "DISTINCT predicate with negation":

a) Subclause 8.14, "<distinct predicate>":

i) Without Feature T152, "DISTINCT predicate with negation", conforming SQL language shall not contain a <distinct predicate part 2> that immediately contains NOT.

147) Specifications for Feature T171, "LIKE clause in table definition":

a) Subclause 11.3, "<table definition>":

i) Without Feature T171, "LIKE clause in table definition", conforming SQL language shall not contain a <like clause>.

148) Specifications for Feature T172, "AS subquery clause in table definition":

a) Subclause 11.3, "<table definition>":

i) Without Feature T172, "AS subquery clause in table definition", conforming SQL language shall not contain an <as subquery clause>.

149) Specifications for Feature T173, "Extended LIKE clause in table definition":

a) Subclause 11.3, "<table definition>":

i) Without Feature T173, "Extended LIKE clause in table definition", a <like clause> shall not contain <like options>.

150) Specifications for Feature T174, "Identity columns":

a) Subclause 11.4, "<column definition>":

i) Without Feature T174, "Identity columns", conforming SQL language shall not contain an <identity column specification>.

b) Subclause 11.17, "<alter identity column specification>":

i) Without Feature T174, "Identity columns", an <alter column definition> shall not contain an <alter identity column specification>.

151) Specifications for Feature T175, "Generated columns":

a) Subclause 11.4, "<column definition>":

i) Without Feature T175, "Generated columns", conforming SQL language shall not contain a <generation clause>.

152) Specifications for Feature T176, "Sequence generator support":

a) Subclause 5.4, "Names and identifiers":

i) Without Feature T176, "Sequence generator support", conforming SQL language shall not contain a <sequence generator name>.

b) Subclause 6.13, "<next value expression>":

i) Without Feature T176, "Sequence generator support", conforming SQL language shall not contain a <next value expression>.

c) Subclause 11.62, "<sequence generator definition>":

i) Without Feature T176, "Sequence generator support", conforming SQL language shall not contain a <sequence generator definition>.

d) Subclause 11.63, "<alter sequence generator statement>":

i) Without Feature T176, "Sequence generator support", conforming SQL language shall not contain an <alter sequence generator statement>.

e) Subclause 11.64, "<drop sequence generator statement>":

i) Without Feature T176, "Sequence generator support", conforming SQL language shall not contain a <drop sequence generator statement>.

153) Specifications for Feature T191, "Referential action RESTRICT":

a) Subclause 11.8, "<referential constraint definition>":

i) Without Feature T191, "Referential action RESTRICT", conforming SQL language shall not contain a <referential action> that contains RESTRICT.

154) Specifications for Feature T201, "Comparable data types for referential constraints":

a) Subclause 11.8, "<referential constraint definition>":

i) Without Feature T201, "Comparable data types for referential constraints", conforming SQL language shall not contain a <referencing columns> in which the data type of each referencing column is not the same as the data type of the corresponding referenced column.

NOTE 481 — The Conformance Rules of Subclause 9.10, "Grouping operations", also apply.

155) Specifications for Feature T211, "Basic trigger capability":

a) Subclause 7.6, "<table reference>":

i) Without Feature T211, "Basic trigger capability", conforming SQL language shall not contain a <transition table name>.

b) Subclause 11.39, "<trigger definition>":

i) Without Feature T211, "Basic trigger capability", conforming SQL language shall not contain a <trigger definition>.

c) Subclause 11.40, "<drop trigger statement>":

i) Without Feature T211, "Basic trigger capability", conforming SQL language shall not contain a <drop trigger statement>.

d) Subclause 12.3, "<privileges>":

i) Without Feature T211, "Basic trigger capability", conforming SQL language shall not contain an <action> that contains TRIGGER.

156) Specifications for Feature T212, "Enhanced trigger capability":

a) Subclause 11.39, "<trigger definition>":

i) Without Feature T212, "Enhanced trigger capability", in conforming SQL language, a <triggered action> shall contain FOR EACH ROW.

157) Specifications for Feature T231, "Sensitive cursors":

a) Subclause 14.1, "<declare cursor>":

i) Without Feature T231, "Sensitive cursors", conforming SQL language shall not contain a <cursor sensitivity> that immediately contains SENSITIVE.

ii) Without Feature F791, "Insensitive cursors", or Feature T231, "Sensitive cursors", conforming SQL language shall not contain a <cursor sensitivity> that immediately contains ASENSITIVE.

158) Specifications for Feature T241, "START TRANSACTION statement":

a) Subclause 16.1, "<start transaction statement>":

i) Without Feature T241, "START TRANSACTION statement", conforming SQL language shall not contain a <start transaction statement>.

159) Specifications for Feature T251, "SET TRANSACTION statement: LOCAL option":

a) Subclause 16.2, "<set transaction statement>":

i) Without Feature T251, "SET TRANSACTION statement: LOCAL option", conforming SQL language shall not contain a <set transaction statement> that immediately contains LOCAL.

160) Specifications for Feature T261, "Chained transactions":

a) Subclause 16.6, "<commit statement>":

i) Without Feature T261, "Chained transactions", conforming SQL language shall not contain a <commit statement> that immediately contains CHAIN.

b) Subclause 16.7, "<rollback statement>":

i) Without Feature T261, "Chained transactions", conforming SQL language shall not contain a <rollback statement> that immediately contains CHAIN.

161) Specifications for Feature T271, "Savepoints":

a) Subclause 5.4, "Names and identifiers":

i) Without Feature T271, "Savepoints", conforming SQL language shall not contain a <savepoint name>.

b) Subclause 16.4, "<savepoint statement>":

i) Without Feature T271, "Savepoints", conforming SQL language shall not contain a <savepoint statement>.

c) Subclause 16.5, "<release savepoint statement>":

i) Without Feature T271, "Savepoints", conforming SQL language shall not contain a <release savepoint statement>.

d) Subclause 16.7, "<rollback statement>":

i) Without Feature T271, "Savepoints", conforming SQL language shall not contain a <savepoint clause>.

162) Specifications for Feature T272, "Enhanced savepoint management":

a) Subclause 11.50, "<SQL-invoked routine>":

i) Without Feature T272, "Enhanced savepoint management", conforming SQL language shall not contain a <routine characteristics> that contains a <savepoint level indication>.

163) Specifications for Feature T281, "SELECT privilege with column granularity":

a) Subclause 12.3, "<privileges>":

i) Without Feature T281, "SELECT privilege with column granularity", in conforming SQL language, an <action> that contains SELECT shall not contain a <privilege column list>.

164) Specifications for Feature T301, "Functional dependencies":

a) Subclause 7.10, "<having clause>":

i) Without Feature T301, "Functional dependencies", in conforming SQL language, each column reference directly contained in the <search condition> shall be one of the following:

1) An unambiguous reference to a grouping column of *T*.

2) An outer reference.

ii) Without Feature T301, "Functional dependencies", in conforming SQL language, each column reference contained in a <subquery> in the <search condition> that references a column of *T* shall be one of the following:

1) An unambiguous reference to a grouping column of *T*.

2) Contained in an aggregated argument of a <set function specification>.

b) Subclause 7.11, "<window clause>":

i) Without Feature T301, "Functional dependencies", in conforming SQL language, if *T* is a grouped table, then each column reference contained in <window clause> that references a column of *T* shall be a reference to a grouping column of *T* or be contained in an aggregated argument of a <set function specification>.

c) Subclause 7.12, "<query specification>":

i) Without Feature T301, "Functional dependencies", in conforming SQL language, if *T* is a grouped table, then in each <value expression> contained in the <select list>, each <column reference> that references a column of *T* shall reference a grouping column or be specified in an aggregated argument of a <set function specification>.

d) Subclause 19.4, "<get descriptor statement>":

i) Without Feature T301, "Functional dependencies", conforming SQL language shall not contain a <descriptor item name> that contains KEY_MEMBER.

165) Specifications for Feature T312, "OVERLAY function":

a) Subclause 6.29, "<string value function>":

i) Without Feature T312, "OVERLAY function", conforming SQL language shall not contain a <character overlay function>.

ii) Without Feature T312, "OVERLAY function", conforming SQL language shall not contain a <blob overlay function>.

166) Specifications for Feature T322, "Overloading of SQL-invoked functions and procedures":

a) Subclause 11.50, "<SQL-invoked routine>":

i) Without Feature T322, "Overloading of SQL-invoked functions and procedures", conforming SQL language shall not contain a <schema routine> in which the schema identified by the explicit or implicit schema name of the <schema qualified routine name> includes a routine descriptor whose routine name is <schema qualified routine name>.

167) Specifications for Feature T323, "Explicit security for external routines":

a) Subclause 11.50, "<SQL-invoked routine>":

i) Without Feature T323, "Explicit security for external routines", conforming SQL language shall not contain an <external security clause>.

168) Specifications for Feature T324, "Explicit security for SQL routines":

a) Subclause 11.50, "<SQL-invoked routine>":

i) Without Feature T324, "Explicit security for SQL routines", conforming SQL language shall not contain a <rights clause>.

169) Specifications for Feature T325, "Qualified SQL parameter references":

a) Subclause 6.6, "<identifier chain>":

i) Without Feature T325, "Qualified SQL parameter references", conforming SQL language shall not contain an SQL parameter reference whose first <identifier> is the <qualified identifier> of a <routine name>.

b) Subclause 7.12, "<query specification>":

i) Without Feature T325, "Qualified SQL parameter references", conforming SQL language shall not contain an <asterisked identifier chain> whose referent is an SQL parameter and whose first <identifier> is the <qualified identifier> of a <routine name>.

170) Specifications for Feature T326, "Table functions":

a) Subclause 6.39, "<multiset value constructor>":

i) Without Feature T326, "Table functions", a <multiset value constructor> shall not contain a <table value constructor by query>.

b) Subclause 7.6, "<table reference>":

i) Without Feature T326, "Table functions", conforming SQL language shall not contain a <table function derived table>.

c) Subclause 11.50, "<SQL-invoked routine>":

i) Without Feature T326, "Table functions", conforming SQL language shall not contain a <returns table type>.

171) Specifications for Feature T331, "Basic roles":

a) Subclause 5.4, "Names and identifiers":

i) Without Feature T331, "Basic roles", conforming SQL language shall not contain a <role name>.

b) Subclause 12.4, "<role definition>":

i) Without Feature T331, "Basic roles", conforming SQL language shall not contain a <role definition>.

c) Subclause 12.5, "<grant role statement>":

i) Without Feature T331, "Basic roles", conforming SQL language shall not contain a <grant role statement>.

d) Subclause 12.6, "<drop role statement>":

i) Without Feature T331, "Basic roles", conforming SQL language shall not contain a <drop role statement>.

e) Subclause 12.7, "<revoke statement>":

i) Without Feature T331, "Basic roles", conforming SQL language shall not contain a <revoke role statement>.

f) Subclause 18.3, "<set role statement>":

i) Without Feature T331, "Basic roles", conforming SQL language shall not contain a <set role statement>.

172) Specifications for Feature T332, "Extended roles":

a) Subclause 6.4, "<value specification> and <target specification>":

i) Without Feature T332, "Extended roles", conforming SQL language shall not contain CURRENT_ROLE.

b) Subclause 11.5, "<default clause>":

i) Without Feature T332, "Extended roles", conforming SQL language shall not contain a <default option> that contains CURRENT_ROLE.

c) Subclause 12.3, "<privileges>":

i) Without Feature T332, "Extended roles", conforming SQL language shall not contain a <grantor>.

d) Subclause 12.4, "<role definition>":

i) Without Feature T332, "Extended roles", conforming SQL language shall not contain a <role definition> that immediately contains WITH ADMIN.

173) Specifications for Feature T351, "Bracketed comments":

a) Subclause 5.2, "<token> and <separator>":

i) Without Feature T351, "Bracketed comments", conforming SQL language shall not contain a <bracketed comment>.

174) Specifications for Feature T431, "Extended grouping capabilities":

a) Subclause 6.9, "<set function specification>":

i) Without Feature T431, "Extended grouping capabilities", conforming SQL language shall not contain a <grouping operation>.

b) Subclause 7.9, "<group by clause>":

i) Without Feature T431, "Extended grouping capabilities", conforming SQL language shall not contain a <rollup list>.

ii) Without Feature T431, "Extended grouping capabilities", conforming SQL language shall not contain a <cube list>.

iii) Without Feature T431, "Extended grouping capabilities", conforming SQL language shall not contain a <grouping sets specification>.

iv) Without Feature T431, "Extended grouping capabilities", conforming SQL language shall not contain an <empty grouping set>.

v) Without Feature T431, "Extended grouping capabilities", conforming SQL language shall not contain an <ordinary grouping set> that contains a <grouping column reference list>.

175) Specifications for Feature T432, "Nested and concatenated GROUPING SETS":

a) Subclause 7.9, "<group by clause>":

i) Without Feature T432, "Nested and concatenated GROUPING SETS", conforming SQL language shall not contain a <grouping set list> that contains a <grouping sets specification>.

ii) Without Feature T432, "Nested and concatenated GROUPING SETS", conforming SQL language shall not contain a <group by clause> that simply contains a <grouping sets specification> *GSS* where *GSS* is not the only <grouping element> simply contained in the <group by clause>.

NOTE 482 — The Conformance Rules of Subclause 9.10, "Grouping operations", also apply.

176) Specifications for Feature T433, "Multiargument GROUPING function":

a) Subclause 6.9, "<set function specification>":

i) Without Feature T433, "Multiargument GROUPING function", conforming SQL language shall not contain a <grouping operation> that contains more than one <column reference>.

177) Specifications for Feature T434, "GROUP BY DISTINCT":

a) Subclause 7.9, "<group by clause>":

i) Without Feature T434, "GROUP BY DISTINCT", conforming SQL language shall not contain a <group by clause> that simply contains a <set quantifier>.

178) Specifications for Feature T441, "ABS and MOD functions":

a) Subclause 6.27, "<numeric value function>":

i) Without Feature T441, "ABS and MOD functions", conforming language shall not contain an <absolute value expression>.

ii) Without Feature T441, "ABS and MOD functions", conforming language shall not contain a <modulus expression>.

179) Specifications for Feature T461, "Symmetric BETWEEN predicate":

a) Subclause 8.3, "<between predicate>":

i) Without Feature T461, "Symmetric BETWEEN predicate", conforming SQL language shall not contain SYMMETRIC or ASYMMETRIC.

NOTE 483 — Since <between predicate> is an ordering operation, the Conformance Rules of Subclause 9.12, "Ordering operations", also apply.

180) Specifications for Feature T471, "Result sets return value":

a) Subclause 11.50, "<SQL-invoked routine>":

i) Without Feature T471, "Result sets return value", conforming SQL language shall not contain a <dynamic result sets characteristic>.

b) Subclause 14.1, "<declare cursor>":

i) Without Feature T471, "Result sets return value", conforming SQL language shall not contain a <cursor returnability>.

181) Specifications for Feature T491, "LATERAL derived table":

a) Subclause 7.6, "<table reference>":

i) Without Feature T491, "LATERAL derived table", conforming SQL language shall not contain a <lateral derived table>.

182) Specifications for Feature T501, "Enhanced EXISTS predicate":

a) Subclause 8.9, "<exists predicate>":

i) Without Feature T501, "Enhanced EXISTS predicate", conforming SQL language shall not contain an <exists predicate> that simply contains a <table subquery> in which the <select list> of a <query specification> directly contained in the <table subquery> does not comprise either an <asterisk> or a single <derived column>.

183) Specifications for Feature T511, "Transaction counts":

a) Subclause 22.1, "<get diagnostics statement>":

i) Without Feature T511, "Transaction counts", conforming SQL language shall not contain a <statement information item name> that contains TRANSACTIONS_COMMITTED, TRANSACTIONS_ROLLED_BACK, or TRANSACTION_ACTIVE.

184) Specifications for Feature T551, "Optional key words for default syntax":

a) Subclause 7.13, "<query expression>":

i) Without Feature T551, "Optional key words for default syntax", conforming SQL language shall not contain UNION DISTINCT, EXCEPT DISTINCT, or INTERSECT DISTINCT.

b) Subclause 14.1, "<declare cursor>":

i) Without Feature T551, "Optional key words for default syntax", conforming SQL language shall not contain a <cursor holdability> that immediately contains WITHOUT HOLD.

185) Specifications for Feature T561, "Holdable locators":

a) Subclause 14.14, "<free locator statement>":

i) Without Feature T561, "Holdable locators", conforming SQL language shall not contain a <free locator statement>.

b) Subclause 14.15, "<hold locator statement>":

i) Without Feature T561, "Holdable locators", conforming SQL language shall not contain a <hold locator statement>.

186) Specifications for Feature T571, "Array-returning external SQL-invoked functions":

a) Subclause 11.41, "<user-defined type definition>":

i) Without Feature T571, "Array-returning external SQL-invoked functions", conforming SQL language shall not contain a <method specification> that contains a <returns clause> that satisfies either of the following conditions:

1) A <result cast from type> is specified that simply contains an <array type> and does not contain a <locator indication>.

2) A <result cast from type> is not specified and <returns data type> simply contains an <array type> and does not contain a <locator indication>.

b) Subclause 11.50, "<SQL-invoked routine>":

i) Without Feature T571, "Array-returning external SQL-invoked functions", conforming SQL language shall not contain an <SQL-invoked routine> that defines an array-returning external function.

187) Specifications for Feature T572, "Multiset-returning external SQL-invoked functions":

a) Subclause 11.41, "<user-defined type definition>":

i) Without Feature T572, "Multiset-returning external SQL-invoked functions", conforming SQL language shall not contain a <method specification> that contains a <returns clause> that satisfies either of the following conditions:

1) A <result cast from type> is specified that simply contains a <multiset type> and does not contain a <locator indication>.

2) A <result cast from type> is not specified and <returns data type> simply contains a <multiset type> and does not contain a <locator indication>.

b) Subclause 11.50, "<SQL-invoked routine>":

i) Without Feature T572, "Multiset-returning external SQL-invoked functions", conforming SQL language shall not contain an <SQL-invoked routine> that defines a multiset-returning external function.

188) Specifications for Feature T581, "Regular expression substring function":

a) Subclause 6.29, "<string value function>":

i) Without Feature T581, "Regular expression substring function", conforming SQL language shall not contain a <regular expression substring function>.

189) Specifications for Feature T591, "UNIQUE constraints of possibly null columns":

a) Subclause 11.7, "<unique constraint definition>":

i) Without Feature T591, "UNIQUE constraints of possibly null columns", in conforming SQL language, if UNIQUE is specified, then the <column definition> for each column whose <column name> is contained in the <unique column list> shall contain NOT NULL.

NOTE 484 — The Conformance Rules of Subclause 9.10, "Grouping operations", also apply.

190) Specifications for Feature T601, "Local cursor references":

a) Subclause 5.4, "Names and identifiers":

i) Without Feature T601, "Local cursor references", a <cursor name> shall not contain a <local qualifier>.

191) Specifications for Feature T611, "Elementary OLAP operations":

a) Subclause 6.10, "<window function>":

i) Without Feature T611, "Elementary OLAP operations", conforming SQL language shall not contain a <window function>.

b) Subclause 7.11, "<window clause>":

i) Without Feature T611, "Elementary OLAP operations", conforming SQL language shall not contain a <window specification>.

c) Subclause 10.10, "<sort specification list>":

i) Without Feature T611, "Elementary OLAP operations", conforming SQL language shall not contain a <null ordering>.

NOTE 485 — The Conformance Rules of Subclause 9.12, "Ordering operations", also apply.

192) Specifications for Feature T612, "Advanced OLAP operations":

a) Subclause 5.4, "Names and identifiers":

i) Without Feature T612, "Advanced OLAP operations", conforming SQL language shall not contain a <window name>.

b) Subclause 6.10, "<window function>":

i) Without Feature T612, "Advanced OLAP operations", conforming SQL language shall not contain a <window name>.

ii) Without Feature T612, "Advanced OLAP operations", conforming SQL language shall not contain PERCENT_RANK or CUME_DIST.

iii) Without Feature T612, "Advanced OLAP operations", conforming SQL language shall not contain a <window function> that simply contains ROW_NUMBER and immediately contains a <window name or specification> whose window structure descriptor does not contain a window ordering clause.

c) Subclause 6.27, "<numeric value function>":

i) Without Feature T612, "Advanced OLAP operations", conforming SQL language shall not contain a <width bucket function>.

d) Subclause 7.11, "<window clause>":

i) Without Feature T612, "Advanced OLAP operations", conforming SQL language shall not contain a <window clause>.

ii) Without Feature T612, "Advanced OLAP operations", conforming SQL language shall not contain an <existing window name>.

iii) Without Feature T612, "Advanced OLAP operations", conforming SQL language shall not contain a <window frame exclusion>.

NOTE 486 — The Conformance Rules of Subclause 9.10, "Grouping operations", also apply.

e) Subclause 10.9, "<aggregate function>":

i) Without Feature T612, "Advanced OLAP operations", conforming SQL language shall not contain a <hypothetical set function> or an <inverse distribution function>.

ii) Without Feature T612, "Advanced OLAP operations", conforming SQL language shall not contain a <filter clause>.

193) Specifications for Feature T613, "Sampling":

a) Subclause 7.6, "<table reference>":

i) Without Feature T613, "Sampling", conforming SQL language shall not contain a <sample clause>.

194) Specifications for Feature T621, "Enhanced numeric functions":

a) Subclause 6.27, "<numeric value function>":

i) Without Feature T621, "Enhanced numeric functions", conforming SQL language shall not contain a <natural logarithm>.

ii) Without Feature T621, "Enhanced numeric functions", conforming SQL language shall not contain an <exponential function>.

iii) Without Feature T621, "Enhanced numeric functions", conforming SQL language shall not contain a <power function>.

iv) Without Feature T621, "Enhanced numeric functions", conforming SQL language shall not contain a <square root>.

v) Without Feature T621, "Enhanced numeric functions", conforming SQL language shall not contain a <floor function>.

vi) Without Feature T621, "Enhanced numeric functions", conforming SQL language shall not contain a <ceiling function>.

b) Subclause 10.9, "<aggregate function>":

i) Without Feature T621, "Enhanced numeric functions", conforming SQL language shall not contain a <computational operation> that immediately contains STDDEV_POP, STDDEV_SAMP, VAR_POP, or VAR_SAMP.

ii) Without Feature T621, "Enhanced numeric functions", conforming SQL language shall not contain a <binary set function type>.

195) Specifications for Feature T641, "Multiple column assignment":

a) Subclause 14.12, "<set clause list>":

i) Without Feature T641, "Multiple column assignment", conforming SQL language shall not contain a <multiple column assignment>.

196) Specifications for Feature T651, "SQL-schema statements in SQL routines":

a) Subclause 11.50, "<SQL-invoked routine>":

i) Without Feature T651, "SQL-schema statements in SQL routines", conforming SQL language shall not contain an <SQL routine body> that contains an SQL-schema statement.

197) Specifications for Feature T652, "SQL-dynamic statements in SQL routines":

a) Subclause 11.50, "<SQL-invoked routine>":

i) Without Feature T652, "SQL-dynamic statements in SQL routines", conforming SQL language shall not contain an <SQL routine body> that contains an SQL-dynamic statement.

198) Specifications for Feature T653, "SQL-schema statements in external routines":

a) Subclause 11.50, "<SQL-invoked routine>":

i) Without Feature T653, "SQL-schema statements in external routines", conforming SQL language shall not contain an <external routine name> that identifies a program in which an SQL-schema statement appears.

199) Specifications for Feature T654, "SQL-dynamic statements in external routines":

a) Subclause 11.50, "<SQL-invoked routine>":

i) Without Feature T654, "SQL-dynamic statements in external routines", conforming SQL language shall not contain an <external routine name> that identifies a program in which an SQL-dynamic statement appears.

200) Specifications for Feature T655, "Cyclically dependent routines":

a) Subclause 11.50, "<SQL-invoked routine>":

i) Without Feature T655, "Cyclically dependent routines", conforming SQL language shall not contain an <SQL routine body> that contains a <routine invocation> whose subject routine is generally dependent on the routine descriptor of the SQL-invoked routine specified by <SQL-invoked routine>.

This page intentionally left blank.

Annex B

(informative)

Implementation-defined elements

This Annex references those features that are identified in the body of this part of ISO/IEC 9075 as implementation-defined.

1) Subclause 4.2.2, "Comparison of character strings":

   a) The specific character set associated with the subtype of character string represented by the <key word>s NATIONAL CHARACTER is implementation-defined.

   b) The circumstances in which conversion of non-UCS character string expressions from one character set to another is automatic is implementation-defined.

2) Subclause 4.2.4, "Character repertoires":

   a) The character repertoires, including standard defined repertoires, supported by the SQL-implementation are implementation-defined.

   b) The character set named SQL_TEXT is an implementation-defined character set that contains every character that is in <SQL language character> and all characters that are in other character sets supported by the SQL-implementation.

   c) The character set named SQL_IDENTIFIER is an implementation-defined character set that contains every character that is in <SQL language character> and all characters that the SQL-implementation supports for use in <regular identifier>s, which is the same as the repertoire that the SQL-implementation supports for use in <delimited identifier>s.

3) Subclause 4.2.5, "Character encoding forms":

   a) The character encodings in the SQL_CHARACTER character encoding form are implementation-defined.

   b) The character encodings in the SQL_TEXT character encoding form are implementation-defined.

   c) The character encodings in the SQL_IDENTIFIER character encoding form are implementation-defined.

   d) Which character encoding forms, including standard defined encoding forms, is implemented are implementation-defined.

4) Subclause 4.2.6, "Collations":

   a) The collations, including standard defined collations, supported by the SQL-implementation are implementation-defined.

   b) The ordering specified by the SQL_CHARACTER collation is implementation-defined.

   c) The ordering specified by the SQL_TEXT collation is implementation-defined.

d) The ordering specified by the SQL_IDENTIFIER collation is implementation-defined.

5) Subclause 4.2.7, "Character sets":

a) It is implementation-defined which collation, UCS_BASIC or UNICODE, is the default for the UTF8, UTF16, and UTF32 character sets.

6) Subclause 4.2.8, "Universal character sets":

a) With the exception of <normalize function> and <normalized predicate>, the result of any operation on an unnormalized UCS string is implementation-defined.

7) Subclause 4.4, "Numbers":

a) Whether truncation or rounding is performed when trailing digits are removed from a numeric value is implementation-defined.

b) When an approximation is obtained by truncation or rounding and there are more than one approximation, then it is implementation-defined which approximation is chosen.

c) It is implementation-defined which numeric values have approximations obtained by rounding or truncation for a given approximate numeric type.

d) The boundaries within which the normal rules of arithmetic apply are implementation-defined.

8) Subclause 4.4.2, "Characteristics of numbers":

a) When converting between numeric data types, if least significant digits are lost, then it is implementation-defined whether rounding or truncation occurs.

9) Subclause 4.6.2, "Datetimes":

a) Whether an SQL-implementation supports leap seconds, and the consequences of such support for date and interval arithmetic, are implementation-defined.

10) Subclause 4.9, "Reference types":

a) In a host variable, a reference type is materialized as an *N*-octet value, where *N* is implementation-defined.

11) Subclause 4.14.6, "Operations involving tables":

a) If a <table reference> contains a <sample clause>, and the <sample clause> contains <repeatable clause>, then repeated executions of that <table reference> return a result table with identical rows for a given <repeat argument>, provided certain implementation-defined conditions are satisfied.

12) Subclause 4.15.3, "Window functions":

a) If PERCENT_RANK is specified, then the declared type of the result is approximate numeric with implementation-defined precision.

b) If CUME_DIST is specified, then the declared type of the result is approximate numeric with implementation-defined precision.

13) Subclause 4.17.3, "Table constraints":

a) The ordering of the lists of referencing column names and referenced column names in a referential constrict descriptor is implementation-defined, but shall be such that corresponding column names occupy corresponding positions in each list.

14) Subclause 4.18, "Functional dependencies":

a) An SQL-implementation may define additional known functional dependencies.

15) Subclause 4.22, "SQL-client modules":

a) The mechanisms by which SQL-client modules are created or destroyed are implementation-defined.

b) The manner in which an association between an SQL-client module and an SQL-agent is defined is implementation-defined.

c) Whether a compilation unit may invoke or transfer control to other compilation units, written in the same or a different programming language is implementation-defined.

16) Subclause 4.23, "Embedded syntax":

a) Whether a portion of the name space is reserved by an implementation for the names of procedures, subroutines, program variables, branch labels, <SQL-client module definition>s, or <externally-invoked procedure>s is implementation-defined; if a portion of the name space is so reserved, the portion reserved is also implementation-defined.

17) Subclause 4.24, "Dynamic SQL concepts":

a) Within an SQL-session, all prepared statements belong to the same implementation-defined <SQL-client module definition> that is different from any other <SQL-client module definition> that exists simultaneously in the environment.

18) Subclause 4.25, "Direct invocation of SQL":

a) The method of invoking <direct SQL statement>s, the method of raising conditions as a result of <direct SQL statement>s, the method of accessing diagnostic information, and the method of returning the results are all implementation-defined.

19) Subclause 4.32, "Cursors":

a) If a sensitive or asensitive holdable cursor is held open for a subsequent SQL-transaction, then whether any significant changes made to SQL-data (by this or any subsequent SQL-transaction in which the cursor is held open) will be visible through that cursor in the subsequent SQL-transaction before that cursor is closed is implementation-defined.

20) Subclause 4.35, "SQL-transactions":

a) It is implementation-defined whether or not the execution of an SQL-data statement is permitted to occur within the same SQL-transaction as the execution of an SQL-schema statement. If it does occur, then the effect on any open cursor or deferred constraint is also implementation-defined.

b) If an SQL-implementation detects unrecoverable errors and implicitly initiates the execution of a <rollback statement>, an exception condition is raised with an *implementation-defined exception code*.

c) It is implementation-defined whether or not the dynamic execution of an <SQL dynamic data statement> is permitted to occur within the same SQL-transaction as the dynamic execution of an SQL-schema

statement. If it does occur, then the effect on any prepared dynamic statement is also implementation-defined.

21) Subclause 4.36, "SQL-connections":

a) It is implementation-defined how an SQL-implementation uses <SQL-server name> to determine the location, identity, and communication protocol required to access the SQL-server and initiate an SQL-session.

22) Subclause 4.37, "SQL-sessions":

a) When an SQL-session is initiated other than through the use of an explicit <connect statement>, then an SQL-session associated with an implementation-defined SQL-server is initiated. The default SQL-server is implementation-defined.

b) The mechanism and rules by which an SQL-implementation determines whether a call to an <externally-invoked procedure> is the last call within the last active SQL-client module is implementation-defined.

c) An SQL-session uses one or more implementation-defined schemas that contain the instances of any global temporary tables, created local temporary tables, or declared local temporary tables within the SQL-session.

d) When an SQL-session is initiated, there is an implementation-defined default time zone used as the current default time zone displacement of the SQL-session.

e) When an SQL-session is initiated other than through the use of an explicit <connect statement>, there is an implementation-defined initial value of the SQL-session user identifier.

f) When an SQL-session is initiated, there is an implementation-defined default catalog whose name is used to effectively qualify all unqualified <schema name>s contained in <preparable statement>s that are dynamically prepared in the current SQL-session through the execution of <prepare statement>s and <execute immediate statement>s.

g) When an SQL-session is initiated, there is an implementation-defined default schema whose name is used to effectively qualify all unqualified <schema qualified name>s contained in <preparable statement>s that are dynamically prepared in the current SQL-session through the execution of <prepare statement>s and <execute immediate statement>s.

h) The value of the current SQL-path before a successful execution of <set path statement> is implementation-defined.

23) Subclause 5.1, "<SQL terminal character>":

a) The end-of-line indicator (<newline>) is implementation-defined.

24) Subclause 5.2, "<token> and <separator>":

a) Equivalence of two <regular identifier>s, or of a <regular identifier> and a <delimited identifier> is determined using an implementation-defined collation that is sensitive to case.

b) When the source character set does not contain <reverse solidus>, the character used as the default <Unicode escape character> is implementation-defined.

25) Subclause 5.3, "<literal>":

a) The <character set name> character set of the used to represent national characters is implementation-defined.

b) The declared type of an <exact numeric literal> is implementation-defined.

c) The declared type of an <approximate numeric literal> is implementation-defined.

26) Subclause 5.4, "Names and identifiers":

a) If a <schema name> contained in a <schema name clause> but not contained in an SQL-client module does not contain a <catalog name>, then an implementation-defined <catalog name> is implicit.

b) If a <schema name> contained in a <module authorization clause> does not contain a <catalog name>, then an implementation-defined <catalog name> is implicit.

c) Those <identifier>s that are valid <authorization identifier>s are implementation-defined.

d) Those <identifier>s that are valid <catalog name>s are implementation-defined.

e) All <transcoding name>s are implementation-defined.

f) If a <schema name> contained in a <preparable statement> that is dynamically prepared in the current SQL-session through the execution of a <prepare statement> or an <execute immediate statement> does not contain a <catalog name>, then the implementation-defined <catalog name> for the SQL-session is implicit.

g) If a <schema qualified name> contained in a <preparable statement> that is dynamically prepared in the current SQL-session through the execution of a <prepare statement> or an <execute immediate statement> does not contain a <schema name>, then the implementation-defined <schema name> for the SQL-session is implicit.

27) Subclause 6.1, "<data type>":

a) The <character set name> associated with NATIONAL CHARACTER is implementation-defined.

b) If a <precision> is omitted, then an implementation-defined <precision> is implicit.

c) The decimal precision of a data type defined as DECIMAL for each value specified by <precision> is implementation-defined.

d) The precisions of data types defined as SMALLINT, INTEGER, and BIGINT are implementation-defined, but all three data types have the same radix.

e) The binary precision of a data type defined as FLOAT for each value specified by <precision> is implementation-defined.

f) The precision of a data type defined as REAL is implementation-defined.

g) The precision of a data type defined as DOUBLE PRECISION is implementation-defined, but greater than that for REAL.

h) For every <data type>, the limits of the <data type> are implementation-defined.

i) The maximum lengths for character string types and binary string types are implementation-defined.

j) If CHARACTER SET is not specified for <character string type>, then the character set is implementation-defined.

k) For the <exact numeric type>s DECIMAL and NUMERIC, the maximum values of <precision> and of <scale> are implementation-defined.

l) The transformation *ENNF*() of an <exact numeric type> to its normal form, to obtain the data type name saved in a numeric data type descriptor, is implementation-defined, though it shall adhere to the following constraints:

   i) For every <exact numeric type> *ENT*, *ENNF*(*ENT*) shall not specify DEC or INT.

      NOTE 487 — The preceding requirement prohibits the function *ENNF* from returning a value that uses the abbreviated spelling of the two data types; the function shall instead return the long versions of DECIMAL or INTEGER.

   ii) For every <exact numeric type> *ENT*, the precision, scale, and radix of *ENNF*(*ENT*) shall be the precision, scale, and radix of *ENT*.

   iii) For every <exact numeric type> *ENT*, *ENNF*(*ENT*) shall be the same as *ENNF*(*ENNF*(*ENT*)).

   iv) For every <exact numeric type> *ENT*, if *ENNF*(*ENT*) specifies DECIMAL, then *ENNF*(*ENT*) shall specify <precision>, and the precision of *ENNF*(*ENT*) shall be the value of the <precision> specified in *ENNF*(*ENT*).

m) For the <approximate numeric type> FLOAT, the maximum value of <precision> is implementation-defined.

n) The transformation *ANNF*() of an <approximate numeric type> to its normal form, to obtain the data type name saved in a numeric data type descriptor, is implementation-defined, though it shall adhere to the following constraints:

   i) For every <approximate numeric type> *ANT*, the precision of *ANNF*(*ANT*) shall be the precision of *ANT*.

   ii) For every <approximate numeric type> *ANT*, *ANNF*(*ANT*) shall be the same as *ANNF*(*ANNF*(*ANT*)).

   iii) For every <exact numeric type> *ANT*, if *ANNF*(*ANT*) specifies FLOAT, then *ANNF*(*ANT*) shall specify <precision>, and the precision of *ANNF*(*ANT*) shall be the value of the <precision> specified in *ANNF*(*ANT*).

o) For the <approximate numeric type>s FLOAT, REAL, and DOUBLE PRECISION, the maximum and minimum values of the exponent are implementation-defined.

p) The maximum value of <time fractional seconds precision> is implementation-defined, but shall not be less than 6.

q) The maximum values of <time precision> and <timestamp precision> for a <datetime type> are the same implementation-defined value.

r) If the maximum cardinality of an <array type> is omitted, then an implementation-defined maximum cardinality is implicit.

28) Subclause 6.4, "<value specification> and <target specification>":

a) Whether the character string of the <value specification>s CURRENT_USER, SESSION_USER, and SYSTEM_USER is variable-length or fixed-length, and its maximum length if it is variable-length or its length if it is fixed-length, are implementation-defined.

b) The value specified by SYSTEM_USER is an implementation-defined string that represents the operating system user who executed the SQL-client module that contains the SQL-statement whose execution caused the SYSTEM_USER <general value specification> to be evaluated.

c) Whether the data type of CURRENT_PATH is fixed-length or variable-length, and its length if it is fixed-length or its maximum length if it is variable-length, are implementation-defined.

d) If a <target specification> or <simple target specification> is assigned a value that is a zero-length character string, then it is implementation-defined whether an exception condition is raised: *data exception — zero-length character string.*

e) In Intermediate SQL, the specific data type of <indicator parameter>s and <indicator variable>s shall be the same implementation-defined data type.

29) Subclause 6.9, "<set function specification>":

a) The precision of the value derived from application of the COUNT function is implementation-defined.

b) The precision of the value derived from application of the SUM function to a declared type of exact numeric is implementation-defined.

c) The precision and scale of the value derived from application of the AVG function to a declared type of exact numeric is implementation-defined.

d) The precision of the value derived from application of the SUM function or AVG function to a data type of approximate numeric is implementation-defined.

30) Subclause 6.12, "<cast specification>":

a) Whether to round or truncate when casting to exact numeric, approximate numeric, datetime, or interval data types is implementation-defined.

31) Subclause 6.26, "<numeric value expression>":

a) When the declared type of both operands of the addition, subtraction, multiplication, or division operator is exact numeric, the declared type of the result is an implementation-defined exact numeric type.

b) When the declared type of both operands of the division operator is exact numeric, the scale of the result is implementation-defined.

c) When the declared type of either operand of an arithmetic operator is approximate numeric, the declared type of the result is an implementation-defined approximate numeric type.

d) Whether to round or truncate when performing division is implementation-defined.

32) Subclause 6.27, "<numeric value function>":

a) The declared type of <position expression> is an implementation-defined exact numeric type with scale 0 (zero).

b) The declared type of <extract expression> is an implementation-defined exact numeric type. If <primary datetime field> specifies SECOND, then the scale is implementation-defined; otherwise, the scale is 0 (zero).

c) The declared type of <length expression> is an implementation-defined exact numeric type with scale 0 (zero).

d) The declared type of the result of <natural logarithm> is an implementation-defined approximate numeric.

e) The declared type of the result of <exponential function> is an implementation-defined approximate numeric.

f) The declared type of the result of <power function> is an implementation-defined approximate numeric.

g) The declared types of the results of <floor function> and of <ceiling function> are implementation-defined exact numeric type with scale 0 (zero) if the declared type of the simply contained <numeric value expression> is exact numeric; otherwise, the declared types of the results are implementation-defined approximate numeric types.

33) Subclause 6.28, "<string value expression>":

a) If the result of the <character value expression> is a zero-length character string, then it is implementation-defined whether an exception condition is raised: *data exception — zero-length character string*.

b) If the character encoding form of <character factor> is UTF8, UTF16, or UTF32, and either of the operands is not normalized, then the result is implementation-defined.

34) Subclause 6.29, "<string value function>":

a) The maximum length of <character transliteration> or <transcoding> is implementation-defined.

b) The character set of the result of a <transcoding> is implementation-defined.

35) Subclause 6.32, "<interval value expression>":

a) The difference of two values of type TIME (with or without time zone) is constrained to be between –24:00:00 and +24:00:00 (excluding each end point); it is implementation-defined which of two non-zero values in this range is the result, although the computation shall be deterministic.

b) When an interval is produced from the difference of two datetimes, the choice of whether to round or truncate is implementation-defined.

c) The result's <interval leading field precision> is implementation-defined, but shall not be less than the <interval leading field precision> of the <interval primary>.

d) The <interval leading field precision> is implementation-defined, but shall be sufficient to represent all interval values with the interval fields and <interval leading field precision> of <interval value expression 1> as well as all interval values with the interval fields and <interval leading field precision> of <interval term 1>.

36) Subclause 7.12, "<query specification>":

a) An SQL-implementation may define additional implementation-defined rules for recognizing known-not-null columns.

37) Subclause 8.5, "<like predicate>":

a) It is implementation-defined which collations can be used as collations for the <like predicate>.

38) Subclause 8.6, "<similar predicate>":

a) It is implementation-defined which collations can be used as collations for the <similar predicate>.

39) Subclause 9.1, "Retrieval assignment":

a) If a value *V* is approximate numeric and a target *T* is exact numeric, then whether the approximation of *V* retrieved into *T* is obtained by rounding or by truncation is implementation-defined.

b) If a value *V* is datetime with a greater precision than a target *T*, then it is implementation-defined whether the approximation of *V* retrieved into *T* is obtained by rounding or truncation.

c) If a value *V* is interval with a greater precision than a target *T*, then it is implementation-defined whether the approximation of *V* retrieved into *T* is obtained by rounding or by truncation.

40) Subclause 9.2, "Store assignment":

a) If a value *V* is approximate numeric and a target *T* is exact numeric, then whether the approximation of *V* stored into *T* is obtained by rounding or by truncation is implementation-defined.

b) If a value *V* is datetime with a greater precision than a target *T*, then it is implementation-defined whether the approximation of *V* stored into *T* is obtained by rounding or truncation.

c) If a value *V* is interval with a greater precision than a target *T*, then it is implementation-defined whether the approximation of *V* stored into *T* is obtained by rounding or by truncation.

41) Subclause 9.3, "Data types of results of aggregations":

a) If all of the data types in *DTS* are exact numeric, then the result data type is exact numeric with implementation-defined precision.

b) If any data type in *DTS* is approximate numeric, then each data type in *DTS* shall be numeric and the result data type is approximate numeric with implementation-defined precision.

42) Subclause 9.22, "Creation of a sequence generator":

a) If <sequence generator maxvalue option> specifies NO MAXVALUE or if <sequence generator max-value option> is not specified, then a <sequence generator max value> that is an implementation-defined <signed numeric literal> of declared type *DT* is implicit.

b) If <sequence generator minvalue option> specifies NO MINVALUE or if <sequence generator minvalue option> is not specified, then a <sequence generator min value> that is an implementation-defined <signed numeric literal> of declared type *DT* is implicit.

43) Subclause 9.23, "Altering a sequence generator":

a) If <sequence generator maxvalue option> specifies NO MAXVALUE, then a <sequence generator max value> that is an implementation-defined <signed numeric literal> of declared type *DT* is implicit.

b) If <sequence generator minvalue option> specifies NO MINVALUE, then a <sequence generator min value> that is an implementation-defined <signed numeric literal> of declared type *DT* is implicit.

44) Subclause 10.1, "<interval qualifier>":

a) The maximum value of <interval leading field precision> is implementation-defined, but shall not be less than 2.

b) The maximum value of <interval fractional seconds precision> is implementation-defined, but shall not be less than 2.

45) Subclause 10.4, "<routine invocation>":

a) If an SQL-invoked routine does not contain SQL, does not possibly read SQL-data, and does not possibly modify SQL-data, then the SQL-session module of the new SQL-session context *RSC* is set to be an implementation-defined module.

b) If P_i is an output SQL parameter, then CPV_i is an implementation-defined value of type T_i.

c) Whether a syntax error occurs if an <SQL routine body> contains an <SQL connection statement> or an <SQL transaction statement> is implementation-defined.

d) When a new SQL-session context *RSC* is created, the current default catalog name, current default unqualified schema name, the current default character set name, the SQL-path of the current SQL-session, the current default time zone displacement of the current SQL-session, and the contents of all SQL dynamic descriptor areas are set to implementation-defined values.

e) If *R* is an external routine, then it is implementation-defined whether the identities of all instances of created local temporary tables that are referenced in the <SQL-client module definition> of *P*, declared local temporary tables that are defined by <temporary table declaration>s that are contained in the <SQL-client module definition> of *P*, and the cursor position of all open cursors that are defined by <declare cursor>s that are contained in the <SQL-client module definition> of *P* are removed from *RSC*.

f) After the completion of *P*, it is implementation-defined whether open cursors declared in the <SQL-client module definition> of *P* are closed and destroyed, whether local temporary tables associated with *RCS* are destroyed, and whether prepared statements prepared by *P* are deallocated.

g) If *R* is an SQL-invoked procedure, then for each SQL parameter that is an output SQL parameter or both an input and output SQL parameter whose corresponding argument was not assigned a value, that corresponding argument is set to an implementation-defined value of the appropriate type.

h) If the external security characteristic of an external SQL-invoked routine is IMPLEMENTATION DEFINED, then the user identifier and role name in the first cell of the authorization stack of the new SQL-session context are implementation-defined.

46) Subclause 10.9, "<aggregate function>":

a) If COUNT is specified, then the declared type of the result is an implementation-defined exact numeric type with scale of 0 (zero).

b) If SUM or AVG is specified, then:

i) If SUM is specified and the declared type of the argument is exact numeric with scale *S*, then the declared type of the result is an implementation-defined exact numeric type with scale *S*.

ii) If AVG is specified and the declared type *DT* of the argument is exact numeric, then the declared type of the result is an implementation-defined exact numeric type with precision not less than the precision of *DT* and scale not less than the scale of *DT*.

iii) If the declared type *DT* of the argument is approximate numeric, then the declared type of the result is an implementation-defined approximate numeric type with precision not less than the precision of *DT*.

c) If VAR_POP or VAR_SAMP is specified, then the declared type of the result is an implementation-defined approximate numeric type. If the declared type of the argument is approximate numeric, then the precision of the result is not less than the precision of the argument.

d) If <binary set function type> is specified, then

Case:

i) If REGR_COUNT is specified, then the declared type of the result is an implementation-defined exact numeric type with scale of 0 (zero).

ii) Otherwise, the declared type of the result is an implementation-defined approximate numeric type. If either argument is approximate numeric, then the precision of the result shall be at least as great as the precision of the approximate numeric argument(s).

e) If <hypothetical set function> is specified, then

Case:

i) If RANK or DENSE_RANK is specified, then the declared type of the result is exact numeric with implementation-defined precision and with scale 0 (zero).

ii) Otherwise, the declared type of the result is approximate numeric with implemenation-defined precision.

f) If the declared type of the <value expression> simply contained in the <sort specification> of an <inverse distribution function> that specifies PERCENTILE_CONT is numeric, then the result type is approximate numeric with implementation-defined precision.

47) Subclause 10.10, "<sort specification list>":

a) If <null ordering> is not specified, then an implementation-defined <null ordering> is implicit. The implementation-defined default for <null ordering> shall not depend on the context outside of <sort specification list>.

48) Subclause 11.1, "<schema definition>":

a) If <schema character set specification> is not specified, then a <schema character set specification> containing an implementation-defined <character set specification> is implicit.

b) If <schema path specification> is not specified, then a <schema path specification> containing an implementation-defined <schema name list> is implicit.

c) If AUTHORIZATION <authorization identifier> is not specified, then an <authorization identifier> equivalent to the implementation-defined <authorization identifier> for the SQL-session is implicit.

d) The privileges necessary to execute the <schema definition> are implementation-defined.

49) Subclause 11.6, "<table constraint definition>":

a) The ordering of the lists of referencing column names and referenced column names in a referential constraint descriptor is implementation-defined, but shall be such that corresponding column names occupy corresponding positions in each list.

50) Subclause 11.33, "<collation definition>":

a) The <existing collation name>s that are supported are implementation-defined.

b) The collation resulting from the specification of EXTERNAL in a <collation definition> may be implementation-defined.

51) Subclause 11.35, "<transliteration definition>":

a) The <existing transliteration name>s that are supported are implementation-defined.

52) Subclause 11.39, "<trigger definition>":

a) It is implementation-defined whether the <triggered SQL statement> shall not generally contain an <SQL transaction statement>, an <SQL connection statement>, an <SQL schema statement>, an <SQL dynamic statement>, or an <SQL session statement>.

b) It is implementation-defined whether the <triggered action> shall not generally contain an <SQL data change statement> or a <routine invocation> whose subject routine is an SQL-invoked routine that possibly modifies SQL-data.

53) Subclause 11.50, "<SQL-invoked routine>":

a) If READS SQL DATA is specified, then it is implementation-defined whether the SQL routine body shall not contain an <SQL procedure statement> *S* that satisfies at least one of the following:

i) *S* is an <SQL data change statement>.

ii) *S* contains a <routine invocation> whose subject routine is an SQL-invoked routine that possibly modifies SQL-data.

iii) *S* contains an <SQL procedure statement> that is an <SQL data change statement>.

b) If CONTAINS SQL is specified, then it is implementation-defined whether the SQL routine body shall not contain an <SQL procedure statement> *S* that satisfies at least one of the following:

i) *S* is an <SQL data statement> other than <free locator statement> and <hold locator statement>.

ii) *S* contains a <routine invocation> whose subject routine is an SQL-invoked routine that possibly modifies SQL-data or possibly reads SQL-data.

iii) *S* contains an <SQL procedure statement> that is an <SQL data statement> other than <free locator statement> and <hold locator statement>.

c) If DETERMINISTIC is specified, then it is implementation-defined whether the <SQL routine body> shall not contain an <SQL procedure statement> that is possibly non-deterministic.

d) It is implementation-defined whether the <SQL routine body> shall not contain an <SQL connection statement>, an <SQL schema statement>, an <SQL dynamic statement>, or an <SQL transaction statement> other than a <savepoint statement>, <release savepoint statement>, or a <rollback statement> that specifies a <savepoint clause>.

54) Subclause 11.62, "<sequence generator definition>":

a) If <sequence generator data type option> is not specified, then an implementation-defined exact numeric type *DT* with scale 0 (zero) is implicit.

55) Subclause 12.4, "<role definition>":

a) The Access Rules are implementation-defined.

56) Subclause 12.7, "<revoke statement>":

a) When loss of the USAGE privilege on a character set causes an SQL-client module to be determined to be a lost module, the impact on that SQL-client module is implementation-defined.

57) Subclause 13.1, "<SQL-client module definition>":

a) If the explicit or implicit <schema name> does not specify a <catalog name>, then an implementation-defined <catalog name> is implicit.

b) If <module path specification> is not specified, then a <module path specification> containing an implementation-defined <schema name list> is implicit.

c) If a <module character set specification> is not specified, then a <module character set specification> that specifies the implementation-defined character set that contains every character that is in <SQL language character> is implicit.

58) Subclause 14.1, "<declare cursor>":

a) Whether null values shall be considered greater than or less than all non-null values in determining the order of rows in a table associated with a <declare cursor> is implementation-defined.

b) Whether an SQL-implementation is able to disallow significant changes that would not be visible through a currently open cursor is implementation-defined.

59) Subclause 14.2, "<open statement>":

a) The extent to which an SQL-implementation may disallow independent changes that are not significant is implementation-defined.

60) Subclause 14.6, "<delete statement: positioned>":

a) The extent to which an SQL-implementation may disallow independent changes that are not significant is implementation-defined.

b) If there is any sensitive cursor *CR* that is open in the SQL-transaction in which this statement is being executed and *CR* has been held into a subsequent SQL-transaction, then whether the change resulting from the successful execution of this statement is made visible to *CR* is implementation-defined.

61) Subclause 14.7, "<delete statement: searched>":

a) The extent to which an SQL-implementation may disallow independent changes that are not significant is implementation-defined.

b) If there is any sensitive cursor *CR* that is open in the SQL-transaction in which this statement is being executed and *CR* has been held into a subsequent SQL-transaction, then whether the change resulting from the successful execution of this statement is made visible to *CR* is implementation-defined.

62) Subclause 14.8, "<insert statement>":

a) The extent to which an SQL-implementation may disallow independent changes that are not significant is implementation-defined.

b) If there is any sensitive cursor *CR* that is open in the SQL-transaction in which this statement is being executed and *CR* has been held into a subsequent SQL-transaction, then whether the change resulting from the successful execution of this statement is made visible to *CR* is implementation-defined.

63) Subclause 14.9, "<merge statement>":

a) The extent to which an SQL-implementation may disallow independent changes that are not significant is implementation-defined.

b) If there is any sensitive cursor *CR* that is open in the SQL-transaction in which this statement is being executed and *CR* has been held into a subsequent SQL-transaction, then whether the change resulting from the successful execution of this statement is made visible to *CR* is implementation-defined.

64) Subclause 14.10, "<update statement: positioned>":

a) The extent to which an SQL-implementation may disallow independent changes that are not significant is implementation-defined.

b) If there is any sensitive cursor *CR* that is open in the SQL-transaction in which this statement is being executed and *CR* has been held into a subsequent SQL-transaction, then whether the change resulting from the successful execution of this statement is made visible to *CR* is implementation-defined.

65) Subclause 14.11, "<update statement: searched>":

a) The extent to which an SQL-implementation may disallow independent changes that are not significant is implementation-defined.

b) If there is any sensitive cursor *CR* that is open in the SQL-transaction in which this statement is being executed and *CR* has been held into a subsequent SQL-transaction, then whether the change resulting from the successful execution of this statement is made visible to *CR* is implementation-defined.

66) Subclause 16.2, "<set transaction statement>":

a) The isolation level that is set for a transaction is an implementation-defined isolation level that will not exhibit any of the phenomena that the explicit or implicit <level of isolation> would not exhibit, as specified in Table 8, "SQL-transaction isolation levels and the three phenomena".

67) Subclause 16.4, "<savepoint statement>":

a) The maximum number of savepoints per SQL-transaction is implementation-defined.

68) Subclause 16.7, "<rollback statement>":

a) The status of any open cursors in any SQL-client module associated with the current SQL-transaction that were opened by that SQL-transaction before the establishment of a savepoint to which a rollback is executed is implementation-defined.

69) Subclause 17.1, "<connect statement>":

a) If <connection user name> is not specified, then an implementation-defined <connection user name> for the SQL-connection is implicit.

b) The initial value of the current role name is the null value.

c) The restrictions on whether or not the <connection user name> shall be identical to the <module authorization identifier> for the SQL-client module that contains the <externally-invoked procedure> that contains the <connect statement> are implementation-defined.

d) If DEFAULT is specified, then the method by which the default SQL-server is determined is implementation-defined.

e) The method by which <SQL-server name> is used to determine the appropriate SQL-server is implementation-defined.

70) Subclause 18.2, "<set session user identifier statement>":

a) Whether or not the <authorization identifier> for the SQL-session can be set to an <authorization identifier> other than the <authorization identifier> of the SQL-session when the SQL-session is started is implementation-defined, as are any restrictions pertaining to such changes.

71) Subclause 18.10, "<set session collation statement>":

a) If no <character set specification> is specified in a <set session collation statement>, then the character sets for which the SQL-session collations are set are implementation-defined.

72) Subclause 19.2, "<allocate descriptor statement>":

a) If WITH MAX <occurrences> is not specified, then an implementation-defined default value for <occurrences> that is greater than 0 (zero) is implicit.

b) The maximum number of SQL descriptor areas and the maximum number of item descriptor areas for a single SQL descriptor area are implementation-defined.

73) Subclause 19.5, "<set descriptor statement>":

a) Restrictions on changing TYPE, LENGTH, OCTET_LENGTH, SCALE, COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME, CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, and CHARACTER_SET_NAME values resulting from the execution of a <describe statement> before execution of an <execute statement>, <dynamic open statement>, or <dynamic fetch statement> are implementation-defined.

74) Subclause 19.6, "<prepare statement>":

a) The Format and Syntax Rules for a <preparable implementation-defined statement> are implementation-defined.

75) Subclause 19.9, "<describe statement>":

a) The character set of the data type of <descriptor name> is implementation-defined.

b) If *SR* does not contain a single routine SQL-invoked *R*, then the values of PARAMETER_MODE, PARAMETER_ORDINAL_POSITION, PARAMETER_SPECIFIC_CATALOG, PARAMETER_SPECIFIC_SCHEMA, and PARAMETER_SPECIFIC_NAME in the descriptor for each <dynamic parameter specification> simply contained in the <call statement> are set to implementation-defined values.

76) Subclause 20.1, "<embedded SQL host program>":

a) If *H* does not contain an <embedded authorization declaration> that specifies SCHEMA, then the <schema name> of the <module authorization clause> of *M* is implementation-defined.

b) If *H* does not contain an <embedded authorization declaration>, then *M* contains a <module authorization clause> that specifies "SCHEMA *SN*", where *SN* is an implementation-defined <schema name>.

c) If an <embedded character set declaration> is not specified, then an <embedded character set declaration> containing an implementation-defined <character set specification> is implicit.

d) Each <allocate cursor statement> is replaced with a host language procedure or subroutine call of an implementation-defined procedure that associates the <dynamic cursor name> with the prepared statement.

e) If an <embedded SQL host program> does not contain an <embedded path specification>, then the implied module contains an implementation-defined <module path specification>.

77) Subclause 20.4, "<embedded SQL C program>":

a) The implicit character set in a <C character variable>, a <C VARCHAR variable>, or a <C CLOB variable>is implementation-defined.

78) Subclause 20.5, "<embedded SQL COBOL program>":

a) The COBOL data description clauses, in addition to the PICTURE, SIGN, USAGE and VALUE clauses, that may appear in a <COBOL variable definition> are implementation-defined.

79) Subclause 20.9, "<embedded SQL PL/I program>":

a) The PL/I data description clauses, in addition to the <PL/I type specification> and the INITIAL clause, that may appear in a <PL/I variable definition> are implementation-defined.

80) Subclause 21.1, "<direct SQL statement>":

a) The <value specification> that represents the null value is implementation-defined.

b) The Format, Syntax Rules, and Access Rules for <direct implementation-defined statement> are implementation-defined.

c) Whether a <direct implementation-defined statement> may be associated with an active transaction is implementation-defined.

d) Whether a <direct implementation-defined statement> initiates a transaction is implementation-defined.

81) Subclause 22.1, "<get diagnostics statement>":

a) The actual length of variable-length character items in the diagnostics area is implementation-defined but not less than 128.

b) The character string value set for CLASS_ORIGIN and SUBCLASS_ORIGIN for an implementation-defined class code or subclass code is implementation-defined, but shall not be 'ISO 9075'.

c) The value of MESSAGE_TEXT is an implementation-defined character string.

d) Negative values of COMMAND_FUNCTION_CODE are implementation-defined and indicate implementation-defined SQL-statements.

82) Subclause 23.1, "SQLSTATE":

a) The character set associated with the class value and subclass value of the SQLSTATE parameter is implementation-defined.

b) The values and meanings for classes and subclasses that begin with one of the <digit>s '5', '6', '7', '8', or '9' or one of the <simple Latin upper case letter>s 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', or 'Z' are implementation-defined. The values and meanings for all subclasses that are associated with implementation-defined class values are implementation-defined.

83) Clause 24, "Conformance":

a) The method of flagging nonconforming SQL language or processing of conforming SQL language is implementation-defined, as is the list of additional <key word>s that may be required by the SQL-implementation.

This page intentionally left blank.

Annex C

(informative)

Implementation-dependent elements

This Annex references those places where this part of ISO/IEC 9075 states explicitly that the actions of a conforming SQL-implementation are implementation-dependent.

1) Subclause 3.1.6, "Definitions provided in Part 2":

   a) Whether two non-null values of user-defined type whose comprison form is RELATIVE or MAP that result in *Unknown* when tested for equality according to the rules of Subclause 8.2, "<comparison predicate>", are distinct or not is implementation-dependent.

2) Subclause 4.1, "Data types":

   a) The physical representation of a value of a data type is implementation-dependent.

   b) The null value or values for each data type is implementation-dependent.

3) Subclause 4.14, "Tables":

   a) Because global temporary table contents are distinct within SQL-sessions, and created local temporary tables are distinct within SQL-client modules within SQL-sessions, the effective <schema name> of the schema in which the global temporary table or the created local temporary table is instantiated is an implementation-dependent <schema name> that may be thought of as having been effectively derived from the <schema name> of the schema in which the global temporary table or created local temporary table is defined and the implementation-dependent SQL-session identifier associated with the SQL-session.

   b) The effective <schema name> of the schema in which the created local temporary table is instantiated may be thought of as being further qualified by a unique implementation-dependent name associated with the SQL-client module in which the created local temporary table is referenced.

4) Subclause 4.14.9, "Windowed tables":

   a) The window name of a window defined implicitly by an <in-line window specification> is implementation-dependent.

5) Subclause 4.30, "Diagnostics area":

   a) The condition area limit is implementation-dependent when not explicitly specified.

   b) The ordering of the information about conditions placed into the diagnostics area is implementation-dependent, except that the first condition in the diagnostics area always corresponds to the condition corresponding to the SQLSTATE value.

   c) The maximum number of diagnostics area in a diagnostics area stack is implementation-defined.

6) Subclause 4.32, "Cursors":

   a) If the <declare cursor> does not contain an <order by clause>, or contains an <order by clause> that does not specify the order of the rows completely, then the rows of the table have an order that is defined only to the extent that the <order by clause> specifies an order and is otherwise implementation-dependent.

   b) The effect on the position and state of an open cursor when an error occurs during the execution of an SQL-statement that identifies the cursor is implementation-dependent.

   c) If an asensitive cursor is open and a change is made to SQL-data from within the same SQL-transaction other than through that cursor, then whether that change will be visible through that cursor before it is closed is implementation-dependent.

7) Subclause 4.34, "Basic security model":

   a) The mapping of <authorization identifier>s to operating system users is implementation-dependent.

8) Subclause 4.35, "SQL-transactions":

   a) The schema definitions that are implicitly read on behalf of executing an SQL-statement are implementation-dependent.

9) Subclause 4.37, "SQL-sessions":

   a) A unique implementation-dependent SQL-session identifier is associated with each SQL-session.

10) Subclause 4.39, "Client-server operation":

   a) The <SQL-client module name> of the SQL-client module that is effectively materialized on an SQL-server is implementation-dependent.

   b) Following the execution of an <SQL procedure statement> by an SQL-server, diagnostic information is passed in an implementation-dependent manner into the SQL-agent's diagnostics area stack in the SQL-client.

   c) The effect on diagnostic information of incompatibilities between the character repertoires supported by the SQL-client and SQL-server environments is implementation-dependent.

11) Subclause 6.7, "<column reference>":

   a) If *QCR* is a group-invariant column reference and the most specific type of *QCR* is character string, datetime with time zone, or a user-defined type, then *QCR* denotes an implementation-dependent value that is not distinct from the value of *C* in every row of a given group of the qualifying query of *QCR*.

12) Subclause 6.9, "<set function specification>":

   a) The maximum or minimum of a set of values of a user-defined type is implementation-dependent if the comparison of at least two values of the set results in *Unknown*.

13) Subclause 6.12, "<cast specification>":

   a) When a multiset is cast to an array type, the order of elements in the result is implementation-dependent.

14) Subclause 6.31, "<datetime value function>":

a) The time of evaluation of the CURRENT_DATE, CURRENT_TIME, and CURRENT_TIMESTAMP functions during the execution of an SQL-statement is implementation-dependent.

15) Subclause 6.32, "<interval value expression>":

a) The start datetime used for converting intervals to scalars for subtraction purposes is implementation-dependent.

16) Subclause 6.36, "<array value constructor>":

a) The order of array elements in the result of an <array value constructor by query> which is not decided by the <order by clause> is implementation-dependent.

17) Subclause 7.1, "<row value constructor>":

a) The names of the fields of a <row value constructor> that specifies a <row value constructor element list> are implementation-dependent.

b) The names of the fields of a <contextually typed row value constructor> are implementation-dependent.

18) Subclause 7.3, "<table value constructor>":

a) The column names of a <table value constructor> or a <contextually typed table value constructor> are implementation-dependent.

19) Subclause 7.9, "<group by clause>":

a) If the declared type of a grouping column is a user-defined type and the comparison of that column for two rows results in *Unknown*, then the assignment of those rows to groups in the result of the <group by clause> is implementation-dependent.

b) When a <search condition> or <value expression> is applied to a group, the value of a group-invariant column reference whose most specific type is character string, datetime with time zone or a user-defined type, and that references a column that is functionally dependent on the grouping columns is implementation-dependent.

20) Subclause 7.11, "<window clause>":

a) If the window ordering clause of a window structure descriptor is absent, then the window ordering is implementation-dependent.

b) The window ordering of peer rows within a window partition is implementation-dependent, but the window ordering shall be the same for all window structure descriptors that are order-equivalent. It shall also be the same for windows *W1* and *W2* if *W1* is the ordering window for *W2*.

21) Subclause 7.12, "<query specification>":

a) When a column is not named by an <as clause> and is not derived from a single column reference, then the name of the column is implementation-dependent.

b) If the <set quantifier> DISTINCT is specified, and the most specific type of a result column is character string, datetime with time zone or a user-defined type, then the precise values retained in that column after eliminating redundant duplicates is implementation-dependent.

22) Subclause 7.13, "<query expression>":

a) If a <simple table> is neither a <query specification> nor an <explicit table>, then the name of each column of the <simple table> is implementation-dependent.

b) If a <query term> immediately contains INTERSECT and the <column name>s of a pair of corresponding columns of the operand tables are not equivalent, then the result column has an implementation-dependent <column name>.

c) If a <query expression body> immediately contains UNION or INTERSECT, and the <column name>s of a pair of corresponding columns of the operand tables are not equivalent, then the result column has an implementation-dependent <column name>.

23) Subclause 8.2, "<comparison predicate>":

a) When the operations MAX, MIN, DISTINCT, and references to a grouping column refer to a variable-length character string, the specific value selected from the set of equal values is implementation-dependent.

24) Subclause 10.4, "<routine invocation>":

a) Each SQL argument A_i in *SAL* is evaluated, in an implementation-dependent order, to obtain a value V_i.

25) Subclause 10.9, "<aggregate function>":

a) If the declared type of the argument of MAX or MIN is a user-defined type and the comparison of two values results in *Unknown*, then the maximum or minimum is implementation-dependent.

26) Subclause 10.10, "<sort specification list>":

a) If PV_i and QV_i are not null and the result of "PV_i <comp op> QV_i" is *Unknown*, then the relative ordering of $PV_i$and QV_i is implementation-dependent.

b) The relative ordering of two rows that are not distinct with respect to the <sort specification> is implementation-dependent.

27) Subclause 11.6, "<table constraint definition>":

a) The <constraint name> of a constraint that does not specify a <constraint name definition> is implementation-dependent.

28) Subclause 11.8, "<referential constraint definition>":

a) The specific value to use for cascading among various values that are not distinct is implementation-dependent.

29) Subclause 11.24, "<domain definition>":

a) The <constraint name> of a constraint that does not specify a <constraint name definition> is implementation-dependent.

30) Subclause 11.33, "<collation definition>":

a) The collation of characters for which a collation is not otherwise specified is implementation-dependent.

31) Subclause 14.1, "<declare cursor>":

a) If a <declare cursor> does not contain an <order by clause>, then the ordering of rows in the table associated with that <declare cursor> is implementation-dependent.

b) If a <declare cursor> contains an <order by clause> and a group of two or more rows in the table associated with that <declare cursor> contain values that

Case:

i) are the same null value, or

ii) compare equal according to Subclause 8.2, "<comparison predicate>"

in all columns specified in the <order by clause>, then the order in which rows in that group are returned is implementation-dependent.

c) The relative ordering of two non-null values of a user-defined type *UDT* whose comparison as determined by the user-defined ordering of *UDT* is *Unknown* is implementation-dependent.

32) Subclause 14.3, "<fetch statement>":

a) The order of assignment to targets in the <fetch target list> of values returned by a <fetch statement>, other than status parameters, is implementation-dependent.

b) If an error occurs during assignment of a value to a target during the execution of a <select statement: single row>, then the values of targets other than status parameters are implementation-dependent.

c) If an exception condition occurs during the assignment of a value to a target, then the values of all targets are implementation-dependent and *CR* remains positioned on the current row.

d) It is implementation-dependent whether *CR* remains positioned on the current row when an exception condition is raised during the derivation of any <derived column>.

33) Subclause 14.5, "<select statement: single row>":

a) The order of assignment to targets in the <select target list> of values returned by a <select statement: single row>, other than status parameters, is implementation-dependent.

b) If the cardinality of the <query expression> is greater than 1 (one), then it is implementation-dependent whether or not values are assigned to the targets identified by the <select target list>.

c) If an error occurs during assignment of a value to a target during the execution of a <select statement: single row>, then the values of targets other than status parameters are implementation-dependent.

34) Subclause 14.8, "<insert statement>":

a) The generation of the value of a derived self-referencing column is implementation-dependent.

35) Subclause 14.9, "<merge statement>":

a) The generation of the value of a derived self-referencing column is implementation-dependent.

36) Subclause 16.2, "<set transaction statement>":

a) If <number of conditions> is not specified, then an implementation-dependent value not less than 1 (one) is implicit.

37) Subclause 17.3, "<disconnect statement>":

a) If ALL is specified, then *L* is a list representing every active SQL-connection that has been established by a <connect statement> by the current SQL-agent and that has not yet been disconnected by a <disconnect statement>, in an implementation-dependent order.

38) Subclause 19.4, "<get descriptor statement>":

a) If an exception condition is raised in a <get descriptor statement>, then the values of all targets specified by <simple target specification 1> and <simple target specification 2> are implementation-dependent.

b) For a <dynamic parameter specification>, the value of UNNAMED is 1 (one) and the value of NAME is implementation-dependent.

c) The value retrieved by a <get descriptor statement> for any field whose value is undefined is implementation-dependent.

39) Subclause 19.5, "<set descriptor statement>":

a) If an exception condition is raised in a <set descriptor statement>, then the values of all elements of the descriptor specified in the <set descriptor statement> are implementation-dependent.

40) Subclause 19.6, "<prepare statement>":

a) The validity of an <extended statement name> value or a <statement name> in an SQL-transaction different from the one in which the statement was prepared is implementation-dependent.

41) Subclause 19.10, "<input using clause>":

a) When a <describe input statement> is used, the values for NAME, DATA, and INDICATOR in the SQL dynamic descriptor area structure is implementation-dependent. If TYPE indicates a character string type or a binary large object type, then the values of SCALE and PRECISION are implementation-dependent. If TYPE indicates an exact or approximate numeric type, then the values of LENGTH and OCTET_LENGTH are implementation-dependent. If TYPE indicates a boolean type, then the values of PRECISION, SCALE, LENGTH, and OCTET_LENGTH are implementation-dependent.

42) Subclause 19.11, "<output using clause>":

a) When a <describe output statement> is executed, the values of DATA and INDICATOR are implementation-dependent. If TYPE indicates a character string type or a binary large object type, then the values of SCALE and PRECISION are implementation-dependent. If TYPE indicates an exact or approximate numeric type, then the values of LENGTH and OCTET_LENGTH are implementation-dependent. If TYPE indicates a boolean type, then the values of PRECISION, SCALE, LENGTH, and OCTET_LENGTH are implementation-dependent.

43) Subclause 20.1, "<embedded SQL host program>":

a) The <SQL-client module name> of the implied <SQL-client module definition> derived from an <embedded SQL host program> is implementation-dependent.

b) If an <embedded SQL host program> does not contain an <embedded authorization declaration>, then the <module authorization identifier> of the implied <SQL-client module definition> derived from the <embedded SQL host program> is implementation-dependent.

c) In each <declare cursor> in the implied <SQL-client module definition> derived from an <embedded SQL host program>, each <embedded variable name> has been replaced consistently with a distinct <host parameter name> that is implementation-dependent.

d) The <procedure name> of each <externally-invoked procedure> in the implied <SQL-client module definition> derived from an <embedded SQL host program> is implementation-dependent.

e) In each <externally-invoked procedure> in the implied <SQL-client module definition> derived from an <embedded SQL host program>, each <embedded variable name> has been replaced consistently with a distinct <host parameter name> that is implementation-dependent.

f) For <SQL procedure statement>s other than <open statement>s, whether one <externally-invoked procedure> in the implied <SQL-client module definition> derived from an <embedded SQL host program> can correspond to more than one <SQL procedure statement> in the <embedded SQL host program> is implementation-dependent.

g) In each <externally-invoked procedure> in the implied <SQL-client module definition> derived from an <embedded SQL host program>, the order of the instances of <host parameter declaration> is implementation-dependent.

44) Subclause 21.1, "<direct SQL statement>":

a) A <commit statement> or a <rollback statement> is executed. If an unrecoverable error has occurred, or if the direct invocation of SQL terminated unexpectedly, or if any constraint is not satisfied, then a <rollback statement> is performed. Otherwise, the choice of which of these SQL-statements to perform is implementation-dependent. The determination of whether a direct invocation of SQL has terminated unexpectedly is implementation-dependent.

45) Subclause 22.1, "<get diagnostics statement>":

a) The value of ROW_COUNT following the execution of an SQL-statement that does not directly result in the execution of a <delete statement: searched>, an <insert statement>, a <merge statement>, or an <update statement: searched> is implementation-dependent.

b) If <condition number> has a value other than 1 (one), then the association between <condition number> values and specific conditions raised during evaluation of the General Rules for that SQL-statement is implementation-dependent.

This page intentionally left blank.

Annex D

(informative)

Deprecated features

It is intended that the following features will be removed at a later date from a revised version of this part of ISO/IEC 9075:

1) The use of the keyword EXCEPTION to mean the same as the keyword CONDITION has been deprecated.

This page intentionally left blank.

Annex E

(informative)

Incompatibilities with ISO/IEC 9075:1999

This edition of this part of ISO/IEC 9075 introduces some incompatibilities with the earlier version of Database Language SQL as specified in ISO/IEC 9075-2:1999.

Except as specified in this Annex, features and capabilities of Database Language SQL are compatible with ISO/IEC 9075-2:1999.

1) In ISO/IEC 9075-2:1999, the regular expression used in the <similar predicate> required a particular set of distinguished characters. In this edition of ISO/IEC 9075, the following additional characters are distinguished characters:

   — <question mark>

   — <left brace>

   — <right brace>

2) ISO/IEC 9075-2:1999 defined data types called BIT and BIT VARYING. These data types have been deleted from this edition of ISO/IEC 9075.

3) In ISO/IEC 9075-2:1999, it was not permitted to use a <routine name> to qualify an <SQL parameter reference>. This gives rise to an incompatibility with ISO/IEC 9075 in the case where a routine has the same name as one of its parameters, the declared type of that parameter is such that it has components that can be referenced using "dot notation", and one of those components has the same name as one of the other parameters.

   For example, if function F has a parameter named P and another parameter named F of type ROW(P INTEGER), then "F.P" can be a reference to either the parameter P or the field P of the parameter F and is thus a syntax error. In ISO/IEC 9075-2:1999, it unambiguously references the field.

4) In ISO/IEC 9075-2:1999, <predicate>s such as "A = B = C" were permitted, although the BNF did not indicate whether to interpret this as left associative ("(A = B) = C") or right associative ("A = (B = C)"). In this edition of ISO/IEC 9075, all <predicate>s are non-associative; for example, "A = B = C" is prohibited, although the explicit syntax "(A = B) = C" or "A = (B = C)" is permitted.

5) In ISO/IEC 9075-2:1999, the <column name> of a <derived column> that is not a column reference and does not specify an <as clause> is implementation-dependent, though subject to the requirement that the <column name> shall not be equivalent to the <column name> of any column, other than itself, of a table referenced by any <table reference> contained in the SQL-statement. This edition of ISO/IEC 9075 has eliminated the latter requirement.

6) In ISO/IEC 9075-2:1999, a <query expression body>, <query term>, or <query primary> could consist of a <joined table>. None of those three elements can consist of a <joined table> in this edition of ISO/IEC 9075.

7) In ISO/IEC 9075-2:1999, the columns of a table created by a <table definition> that contains a <like clause> inherit only the column names and the data types of the columns of the original table. In this edition of ISO/IEC 9075, the columns also inherit the nullability charactistics of the columns of the original table.

8) In ISO/IEC 9075-2:1999 and ISO/IEC 9075-4:1999, a <savepoint statement> that specifies the name of an existing savepoint *SP*, executed by either of the following, causes *SP* to be destroyed:

   — An atomic <compound statement> (see ISO/IEC 9075-4).

   — A <triggered SQL statement>.

   In this edition of ISO/IEC 9075, *SP* is destroyed only if it exists in the current savepoint level.

9) In ISO/IEC 9075-2:1999 and ISO/IEC 9075-4:1999, a <release savepoint statement> specifying savepoint name *SP*, executed by either of the following, causes *SP* to be destroyed:

   — An atomic <compound statement>.

   — A <triggered SQL statement>.

   In this edition of ISO/IEC 9075, *SP* is destroyed only if *SP* was established in the current savepoint level.

10) In ISO/IEC 9075-2:1999, <column options> could contain a <collate clause>. This functionality has been removed.

11) In ISO/IEC 9075-2:1999, the Syntax Rules of Subclause 11.4, "<column definition>", permitted a <column definition> to contains both a <domain name> and a <collate clause>. That functionality was not satisfactorily specified and has been removed from this edition of ISO/IEC 9075.

12) In ISO/IEC 9075-2:1999, the Identifiers associated with SQL-statement codes 54 and 55 in Table 31, "SQL-statement codes", were "DYNAMIC DELETE CURSOR" and "DYNAMIC UPDATE CURSOR", respectively. In this edition of ISO/IEC 9075, to better distinguish those two SQL-statement codes, the Identifiers have been changed to "PREPARABLE DYNAMIC DELETE CURSOR" and "PREPARABLE DYNAMIC UPDATE CURSOR", respectively.

13) In ISO/IEC 9075-2:1999, a <case abbreviation> could contain a <value expression> that contains a <routine invocation> whose subject routine is an SQL-invoked routine that is possibly non-deterministic or that possibly modifies SQL-data. In this edition of ISO/IEC 9075, that capability has been removed.

14) In ISO/IEC 9075-2:1999, <joined table> contained a UNION JOIN alternative. This edition of ISO/IEC 9075 removes that alternative.

15) ISO/IEC 9075-2:1999 defined the character represented by the Unicode code point U+FEFF (Zero Width No-Break Space) as being one of those used to separate tokens. U+FEFF is not so defined in this edition of ISO/IEC 9075.

16) In ISO/IEC 9075:1999, <field definition>, <column definition>, and <attribute definition> could contain <reference scope check>. That functionality was not satisfactorily specified and has been removed from this edition of ISO/IEC 9075.

17) A number of additional <reserved word>s have been added to the language. These <reserved word>s are:

   — ATOMIC

   — BIGINT

— COLLECT
— CONDITION
— ELEMENT
— FUSION
— INTERSECTION
— MEMBER
— MERGE
— MULTISET
— NORMALIZE
— SUBMULTISET
— TABLESAMPLE
— UESCAPE

This page intentionally left blank.

Annex F

(informative)

SQL feature taxonomy

This Annex describes a taxonomy of features and packages defined in this part of ISO/IEC 9075.

Table 35, "Feature taxonomy and definition for mandatory features", contains a taxonomy of the mandatory features of SQL language that are specified in this part of ISO/IEC 9075. Table 36, "Feature taxonomy for optional features", contains a taxonomy of the optional features of the SQL language that are specified in this part of ISO/IEC 9075.

In these tables, the first column contains a counter that may be used to quickly locate rows of the table; these values otherwise have no use and are not stable — that is, they are subject to change in future editions of or even Technical Corrigenda to ISO/IEC 9075 without notice.

The "Feature ID" column of Table 35, "Feature taxonomy and definition for mandatory features", and of Table 36, "Feature taxonomy for optional features", specifies the formal identification of each feature and each subfeature contained in the table.

The "Feature Name" column of Table 35, "Feature taxonomy and definition for mandatory features", contains a brief description of the feature or subfeature associated with the Feature ID value.

The "Feature Description" column of Table 35, "Feature taxonomy and definition for mandatory features", provides the only definition of the mandatory features of this part of ISO/IEC 9075. This definition consists of indications of specific language elements supported in each feature, subject to the constraints of all Syntax Rules, Access Rules, and Conformance Rules.

Table 35 — Feature taxonomy and definition for mandatory features

	Feature ID	Feature Name	Feature Description
1	**E011**	**Numeric data types**	Subclause 6.1, "<data type>", <numeric type>, including numeric expressions, numeric literals, numeric comparisons, and numeric assignments
2	E011-01	INTEGER and SMALLINT data types (including all spellings)	— Subclause 5.2, "<token> and <separator>": The <reserved word>s INT, INTEGER, and SMALLINT — Subclause 5.3, "<literal>": [<sign>] <unsigned integer> — Subclause 6.1, "<data type>": The INTEGER and SMALLINT <exact numeric type>s — Subclause 13.6, "Data type correspondences": Type correspondences for INTEGER and SMALLINT for all supported languages

	Feature ID	Feature Name	Feature Description
3	E011-02	REAL, DOUBLE PRECISON, and FLOAT data types	— Subclause 5.2, "<token> and <separator>": The <reserved word>s REAL, DOUBLE, FLOAT, and PRECISION — Subclause 5.3, "<literal>": [<sign>] <approximate numeric literal> — Subclause 6.1, "<data type>": <approximate numeric type> — Subclause 13.6, "Data type correspondences": Type correspondences for REAL, DOUBLE PRECISION, and FLOAT for all supported languages
4	E011-03	DECIMAL and NUMERIC data types	— Subclause 5.2, "<token> and <separator>": The <reserved word>s DEC, DECIMAL, and NUMERIC — Subclause 5.3, "<literal>": [<sign>] <exact numeric literal> — Subclause 6.1, "<data type>": The DECIMAL and NUMERIC <exact numeric type>s — Subclause 13.6, "Data type correspondences": Type correspondences for DECIMAL and NUMERIC for all supported languages
5	E011-04	Arithmetic operators	— Subclause 6.26, "<numeric value expression>": When the <numeric primary> is a <value expression primary>
6	E011-05	Numeric comparison	— Subclause 8.2, "<comparison predicate>": For the numeric data types, without support for <table subquery> and without support for Feature F131, "Grouped operations"
7	E011-06	Implicit casting among the numeric data types	— Subclause 8.2, "<comparison predicate>": Values of any of the numeric data types can be compared to each other; such values are compared with respect to their algebraic values — Subclause 9.1, "Retrieval assignment", and Subclause 9.2, "Store assignment": Values of one numeric type can be assigned to another numeric type, subject to rounding, truncation, and out of range conditions
8	**E021**	**Character string types**	— Subclause 6.1, "<data type>": <character string type>, including character expressions, character literals, character comparisons, character assignments, and other operations on character data

	Feature ID	Feature Name	Feature Description
9	E021-01	CHARACTER data type (including all its spellings)	— Subclause 5.2, "<token> and <separator>": The <reserved word>s CHAR and CHARACTER — Subclause 6.1, "<data type>": The CHARACTER <character string type> — Subclause 6.28, "<string value expression>": For values of type CHARACTER — Subclause 13.6, "Data type correspondences": Type correspondences for CHARACTER for all supported languages
10	E021-02	CHARACTER VARYING data type (including all its spellings)	— Subclause 5.2, "<token> and <separator>": The <reserved word>s VARCHAR and VARYING — Subclause 6.1, "<data type>": The CHARACTER VARYING <character string type> — Subclause 6.28, "<string value expression>": For values of type CHARACTER VARYING — Subclause 13.6, "Data type correspondences": Type correspondences for CHARACTER VARYING for all supported languages
11	E021-03	Character literals	— Subclause 5.3, "<literal>": `<quote> [ <character representation>... ] <quote>`
12	E021-04	CHARACTER_LENGTH function	— Subclause 6.27, "<numeric value function>": The <char length expression>
13	E021-05	OCTET_LENGTH function	— Subclause 6.27, "<numeric value function>": The <octet length expression>
14	E021-06	SUBSTRING function	— Subclause 6.29, "<string value function>": The <character substring function>
15	E021-07	Character concatenation	— Subclause 6.28, "<string value expression>": The <concatenation> expression
16	E021-08	UPPER and LOWER functions	— Subclause 6.29, "<string value function>": The <fold> function
17	E021-09	TRIM function	— Subclause 6.29, "<string value function>": The <trim function>

	Feature ID	Feature Name	Feature Description
18	E021-10	Implicit casting among the fixed-length and variable-length character string types	— Subclause 8.2, "<comparison predicate>": Values of either the CHARACTER or CHARACTER VARYING data types can be compared to each other — Subclause 9.1, "Retrieval assignment", and Subclause 9.2, "Store assignment": Values of either the CHARACTER or CHARACTER VARYING data type can be assigned to the other type, subject to truncation conditions
19	E021-11	POSITION function	— Subclause 6.27, "<numeric value function>": The <position expression>
20	E021-12	Character comparison	— Subclause 8.2, "<comparison predicate>": For the CHARACTER and CHARACTER VARYING data types, without support for <table subquery> and without support for Feature F131, "Grouped operations"
21	**E031**	**Identifiers**	— Subclause 5.2, "<token> and <separator>": <regular identifier> and <delimited identifier>
22	E031-01	Delimited identifiers	— Subclause 5.2, "<token> and <separator>": <delimited identifier>
23	E031-02	Lower case identifiers	— Subclause 5.2, "<token> and <separator>": An alphabetic character in a <regular identifier> can be either lower case or upper case (meaning that non-delimited identifiers need not comprise only upper case letters)
24	E031-03	Trailing underscore	— Subclause 5.2, "<token> and <separator>": The list <identifier part> in a <regular identifier> can be an <underscore>
25	**E051**	**Basic query specification**	— Subclause 7.12, "<query specification>": When <table reference> is a <table or query name> that is a <table name>, without the support of Feature F131, "Grouped operations"
26	E051-01	SELECT DISTINCT	— Subclause 7.12, "<query specification>": With a <set quantifier> of DISTINCT, but without subfeatures E051-02 through E051-09

	Feature ID	Feature Name	Feature Description
27	E051-02	GROUP BY clause	— Subclause 7.4, "<table expression>": <group by clause>, but without subfeatures E051-04 through E051-09 — Subclause 7.9, "<group by clause>": With the restrictions that the <group by clause> shall contain all non-aggregated columns in the <select list> and that any column in the <group by clause> shall also appear in the <select list>
28	E051-04	GROUP BY can contain columns not in <select list>	— Subclause 7.9, "<group by clause>": Without the restriction that any column in the <group by clause> shall also appear in the <select list>
29	E051-05	Select list items can be renamed	— Subclause 7.12, "<query specification>": <as clause>
30	E051-06	HAVING clause	— Subclause 7.4, "<table expression>": <having clause> — Subclause 7.10, "<having clause>"
31	E051-07	Qualified * in select list	— Subclause 7.12, "<query specification>": <qualified asterisk>
32	E051-08	Correlation names in the FROM clause	— Subclause 7.6, "<table reference>": [AS] <correlation name>
33	E051-09	Rename columns in the FROM clause	— Subclause 7.6, "<table reference>": [AS] <correlation name> [<left paren> <derived column list> <right paren>]
34	**E061**	**Basic predicates and search conditions**	— Subclause 8.19, "<search condition>", and Subclause 8.1, "<predicate>"
35	E061-01	Comparison predicate	— Subclause 8.2, "<comparison predicate>": For supported data types, without support for <table subquery>
36	E061-02	BETWEEN predicate	— Subclause 8.3, "<between predicate>"
37	E061-03	IN predicate with list of values	— Subclause 8.4, "<in predicate>": Without support for <table subquery>
38	E061-04	LIKE predicate	— Subclause 8.5, "<like predicate>": Without [ESCAPE <escape character>]
39	E061-05	LIKE predicate: ESCAPE clause	— Subclause 8.5, "<like predicate>": With [ESCAPE <escape character>]

	Feature ID	Feature Name	Feature Description
40	E061-06	NULL predicate	— Subclause 8.7, "<null predicate>": Without Feature F481, "Expanded NULL predicate"
41	E061-07	Quantified comparison predicate	— Subclause 8.8, "<quantified comparison predicate>": Without support for <table subquery>
42	E061-08	EXISTS predicate	— Subclause 8.9, "<exists predicate>"
43	E061-09	Subqueries in comparison predicate	— Subclause 8.2, "<comparison predicate>": For supported data types, with support for <table subquery>
44	E061-11	Subqueries in IN predicate	— Subclause 8.4, "<in predicate>": With support for <table subquery>
45	E061-12	Subqueries in quantified comparison predicate	— Subclause 8.8, "<quantified comparison predicate>": With support for <table subquery>
46	E061-13	Correlated subqueries	— Subclause 8.1, "<predicate>": When a <correlation name> can be used in a <table subquery> as a correlated reference to a column in the outer query
47	E061-14	Search condition	— Subclause 8.19, "<search condition>"
48	**E071**	**Basic query expressions**	— Subclause 7.13, "<query expression>"
49	E071-01	UNION DISTINCT table operator	— Subclause 7.13, "<query expression>": With support for `UNION [ DISTINCT ]`
50	E071-02	UNION ALL table operator	— Subclause 7.13, "<query expression>": With support for `UNION ALL`
51	E071-03	EXCEPT DISTINCT table operator	— Subclause 7.13, "<query expression>": With support for `EXCEPT [ DISTINCT ]`
52	E071-05	Columns combined via table operators need not have exactly the same data type.	— Subclause 7.13, "<query expression>": Columns combined via UNION and EXCEPT need not have exactly the same data type
53	E071-06	Table operators in subqueries	— Subclause 7.13, "<query expression>": <table subquery>s can specify UNION and EXCEPT
54	**E081**	**Basic Privileges**	— Subclause 12.3, "<privileges>"
55	E081-01	SELECT privilege at the table level	— Subclause 12.3, "<privileges>": With <action> of SELECT without <privilege column list>

	Feature ID	Feature Name	Feature Description
56	E081-02	DELETE privilege	— Subclause 12.3, "<privileges>": With <action> of DELETE
57	E081-03	INSERT privilege at the table level	— Subclause 12.3, "<privileges>": With <action> of INSERT without <privilege column list>
58	E081-04	UPDATE privilege at the table level	— Subclause 12.3, "<privileges>": With <action> of UPDATE without <privilege column list>
59	E081-05	UPDATE privilege at the column level	— Subclause 12.3, "<privileges>": With <action> of `UPDATE <left paren> <privilege column list> <right paren>`
60	E081-06	REFERENCES privilege at the table level	— Subclause 12.3, "<privileges>": with <action> of REFERENCES without <privilege column list>
61	E081-07	REFERENCES privilege at the column level	— Subclause 12.3, "<privileges>": With <action> of `REFERENCES <left paren> <privilege column list> <right paren>`
62	E081-08	WITH GRANT OPTION	— Subclause 12.2, "<grant privilege statement>": `WITH GRANT OPTION`
63	E081-09	USAGE privilege	— Subclause 12.3, "<privileges>": With <action> of USAGE
64	E081-10	EXECUTE privilege	— Subclause 12.3, "<privileges>": With <action> of EXECUTE
65	**E091**	**Set functions**	— Subclause 6.9, "<set function specification>"
66	E091-01	AVG	— Subclause 6.9, "<set function specification>": With <computational operation> of AVG
67	E091-02	COUNT	— Subclause 6.9, "<set function specification>": With <computational operation> of COUNT
68	E091-03	MAX	— Subclause 6.9, "<set function specification>": With <computational operation> of MAX
69	E091-04	MIN	— Subclause 6.9, "<set function specification>": With <computational operation> of MIN
70	E091-05	SUM	— Subclause 6.9, "<set function specification>": With <computational operation> of SUM

	Feature ID	Feature Name	Feature Description
71	E091-06	ALL quantifier	— Subclause 6.9, "<set function specification>": With <set quantifier> of ALL
72	E091-07	DISTINCT quantifier	— Subclause 6.9, "<set function specification>": With <set quantifier> of DISTINCT
73	**E101**	**Basic data manipulation**	— Clause 14, "Data manipulation": <insert statement>, <delete statement: searched>, and <update statement: searched>
74	E101-01	INSERT statement	— Subclause 14.8, "<insert statement>": When a <contextually typed table value constructor> can consist of no more than a single <contextually typed row value expression>
75	E101-03	Searched UPDATE statement	— Subclause 14.11, "<update statement: searched>": But without support either of Feature E153, "Updatable tables with subqueries", or Feature F221, "Explicit defaults"
76	E101-04	Searched DELETE statement	— Subclause 14.7, "<delete statement: searched>"
77	**E111**	**Single row SELECT statement**	— Subclause 14.5, "<select statement: single row>": Without support of Feature F131, "Grouped operations"
78	**E121**	**Basic cursor support**	— Clause 14, "Data manipulation": <declare cursor>, <open statement>, <fetch statement>, <close statement>, <delete statement: positioned>, and <update statement: positioned>
79	E121-01	DECLARE CURSOR	— Subclause 14.1, "<declare cursor>": When each <value expression> in the <sort key> shall be a <column reference> and that <column reference> shall also be in the <select list>, and <cursor holdability> is not specified
80	E121-02	ORDER BY columns need not be in select list	— Subclause 14.1, "<declare cursor>": Extend subfeature E121-01 so that <column reference> need not also be in the <select list>
81	E121-03	Value expressions in ORDER BY clause	— Subclause 14.1, "<declare cursor>": Extend subfeature E121-01 so that the <value expression> in the <sort key> need not be a <column reference>
82	E121-04	OPEN statement	— Subclause 14.2, "<open statement>"

	Feature ID	Feature Name	Feature Description
83	E121-06	Positioned UPDATE statement	— Subclause 14.10, "<update statement: positioned>": Without support of either Feature E153, "Updateable tables with subqueries" or Feature F221, "Explicit defaults"
84	E121-07	Positioned DELETE statement	— Subclause 14.6, "<delete statement: positioned>"
85	E121-08	CLOSE statement	— Subclause 14.4, "<close statement>"
86	E121-10	FETCH statement: implicit NEXT	— Subclause 14.3, "<fetch statement>"
87	E121-17	WITH HOLD cursors	— Subclause 14.1, "<declare cursor>": Where the <value expression> in the <sort key> need not be a <column reference> and need not be in the <select list>, and <cursor holdability> may be specified
88	**E131**	**Null value support (nulls in lieu of values)**	— Subclause 4.13, "Columns, fields, and attributes": Nullability characteristic — Subclause 6.5, "<contextually typed value specification>": <null specification>
89	**E141**	**Basic integrity constraints**	— Subclause 11.6, "<table constraint definition>": As specified by the subfeatures of this feature in this table
90	E141-01	NOT NULL constraints	— Subclause 11.4, "<column definition>": With <column constraint> of NOT NULL
91	E141-02	UNIQUE constraints of NOT NULL columns	— Subclause 11.4, "<column definition>": With <unique specification> of UNIQUE for columns specified as NOT NULL — Subclause 11.7, "<unique constraint definition>": With <unique specification> of UNIQUE
92	E141-03	PRIMARY KEY constraints	— Subclause 11.4, "<column definition>": With <unique specification> of PRIMARY KEY for columns specified as NOT NULL — Subclause 11.7, "<unique constraint definition>": With <unique specification> of PRIMARY KEY

	Feature ID	Feature Name	Feature Description
93	E141-04	Basic FOREIGN KEY constraint with the NO ACTION default for both referential delete action and referential update action.	— Subclause 11.4, "<column definition>": With <column constraint> of <references specification> — Subclause 11.8, "<referential constraint definition>": Where the columns in the <column name list>, if specified, shall be in the same order as the names in the <unique column list> of the applicable <unique constraint definition> and the <data type>s of the matching columns shall be the same
94	E141-06	CHECK constraints	— Subclause 11.4, "<column definition>": With <column constraint> of <check constraint definition> — Subclause 11.9, "<check constraint definition>"
95	E141-07	Column defaults	— Subclause 11.4, "<column definition>": With <default clause>
96	E141-08	NOT NULL inferred on PRIMARY KEY	— Subclause 11.4, "<column definition>", and Subclause 11.7, "<unique constraint definition>": Remove the restriction in subfeatures E141-02 and E141-03 that NOT NULL be specified along with every PRIMARY KEY and UNIQUE constraint — Subclause 11.4, "<column definition>": NOT NULL is implicit on PRIMARY KEY constraints
97	E141-10	Names in a foreign key can be specified in any order	— Subclause 11.4, "<column definition>", and Subclause 11.8, "<referential constraint definition>": Extend subfeature E141-04 so that the columns in the <column name list>, if specified, need not be in the same order as the names in the <unique column list> of the applicable <unique constraint definition>
98	**E151**	**Transaction support**	— Clause 16, "Transaction management": <commit statement> and <rollback statement>
99	E151-01	COMMIT statement	— Subclause 16.6, "<commit statement>"
100	E151-02	ROLLBACK statement	— Subclause 16.7, "<rollback statement>"
101	**E152**	**Basic SET TRANSACTION statement**	— Subclause 16.2, "<set transaction statement>"
102	E152-01	SET TRANSACTION statement: ISOLATION LEVEL SERIALIZABLE clause	— Subclause 16.2, "<set transaction statement>": With <transaction mode> of ISOLATION LEVEL SERIALIZABLE clause

	Feature ID	Feature Name	Feature Description
103	E152-02	SET TRANSACTION statement: READ ONLY and READ WRITE clauses	— Subclause 16.2, "<set transaction statement>": with <transaction access mode> of READ ONLY or READ WRITE
104	**E153**	**Updatable queries with subqueries**	— Subclause 7.13, "<query expression>": A <query expression> is updatable even though its <where clause> contains a <subquery>
105	**E161**	**SQL comments using leading double minus**	— Subclause 5.2, "<token> and <separator>": <simple comment>
106	**E171**	**SQLSTATE support**	— Subclause 23.1, "SQLSTATE"
107	**E182**	**Module language**	— Clause 13, "SQL-client modules" NOTE 488 — An SQL-implementation is required to supply at least one binding to a standard host language using either module language, embedded SQL, or both.
108	**F031**	**Basic schema manipulation**	— Clause 11, "Schema definition and manipulation": Selected facilities as indicated by the subfeatures of this Feature
109	F031-01	CREATE TABLE statement to create persistent base tables	— Subclause 11.3, "<table definition>": Not in the context of a <schema definition>
110	F031-02	CREATE VIEW statement	— Subclause 11.22, "<view definition>": Not in the context of a <schema definition>, and without support of Feature F081, "UNION and EXCEPT in views"
111	F031-03	GRANT statement	— Subclause 12.1, "<grant statement>": Not in the context of a <schema definition>
112	F031-04	ALTER TABLE statement: ADD COLUMN clause	— Subclause 11.10, "<alter table statement>": The <add column definition> clause — Subclause 11.11, "<add column definition>"
113	F031-13	DROP TABLE statement: RESTRICT clause	— Subclause 11.21, "<drop table statement>": With a <drop behavior> of RESTRICT
114	F031-16	DROP VIEW statement: RESTRICT clause	— Subclause 11.23, "<drop view statement>": With a <drop behavior> of RESTRICT
115	F031-19	REVOKE statement: RESTRICT clause	— Subclause 12.7, "<revoke statement>": With a <drop behavior> of RESTRICT, only where the use of this statement can be restricted to the owner of the table being dropped

	Feature ID	Feature Name	Feature Description
116	**F041**	**Basic joined table**	— Subclause 7.7, "<joined table>"
117	F041-01	Inner join (but not necessarily the INNER keyword)	— Subclause 7.6, "<table reference>": The <joined table> clause, but without support for subfeatures F041-02 through F041-08
118	F041-02	INNER keyword	— Subclause 7.7, "<joined table>": <join type> of INNER
119	F041-03	LEFT OUTER JOIN	— Subclause 7.7, "<joined table>": <outer join type> of LEFT
120	F041-04	RIGHT OUTER JOIN	— Subclause 7.7, "<joined table>": <outer join type> of RIGHT
121	F041-05	Outer joins can be nested	— Subclause 7.7, "<joined table>": Subfeature F041-1 extended so that a <table reference> within the <joined table> can itself be a <joined table>
122	F041-07	The inner table in a left or right outer join can also be used in an inner join	— Subclause 7.7, "<joined table>": Subfeature F041-1 extended so that a <table name> within a nested <joined table> can be the same as a <table name> in an outer <joined table>
123	F041-08	All comparison operators are supported (rather than just =)	— Subclause 7.7, "<joined table>": Subfeature F041-1 extended so that the <join condition> is not limited to a <comparison predicate> with a <comp op> of <equals operator>
124	**F051**	**Basic date and time**	— Subclause 6.1, "<data type>": <datetime type> including datetime literals, datetime comparisons, and datetime conversions
125	F051-01	DATE data type (including support of DATE literal)	— Subclause 5.3, "<literal>": The <date literal> form of <datetime literal> — Subclause 6.1, "<data type>": The DATE <datetime type> — Subclause 6.30, "<datetime value expression>": For values of type DATE

	Feature ID	Feature Name	Feature Description
126	F051-02	TIME data type (including support of TIME literal) with fractional seconds precision of at least 0.	— Subclause 5.3, "<literal>": The <time literal> form of <datetime literal>, where the value of <unquoted time string> is simply <time value> that does not include the optional <time zone interval> — Subclause 6.1, "<data type>": The TIME <datetime type> without the <with or without time zone> clause — Subclause 6.30, "<datetime value expression>": For values of type TIME
127	F051-03	TIMESTAMP data type (including support of TIMESTAMP literal) with fractional seconds precision of at least 0 and 6.	— Subclause 5.3, "<literal>": The <timestamp literal> form of <datetime literal>, where the value of <unquoted timestamp string> is simply <time value> that does not include the optional <time zone interval> — Subclause 6.1, "<data type>": The TIMESTAMP <datetime type> without the <with or without time zone> clause — Subclause 6.30, "<datetime value expression>": For values of type TIMESTAMP
128	F051-04	Comparison predicate on DATE, TIME, and TIMESTAMP data types	— Subclause 8.2, "<comparison predicate>": For comparison between values of the following types: DATE and DATE, TIME and TIME, TIMESTAMP and TIMESTAMP, DATE and TIMESTAMP, and TIME and TIMESTAMP
129	F051-05	Explicit CAST between datetime types and character string types	— Subclause 6.12, "<cast specification>": If support for Feature F201, "CAST function" is available, then CASTing between the following types: from character string to DATE, TIME, and TIMESTAMP; from DATE to DATE, TIMESTAMP, and character string; from TIME to TIME, TIMESTAMP, and character string; from TIMESTAMP to DATE, TIME, TIMESTAMP, and character string
130	F051-06	CURRENT_DATE	— Subclause 6.31, "<datetime value function>": The <current date value function> — Subclause 6.30, "<datetime value expression>": When the value is a <current date value function>

	Feature ID	Feature Name	Feature Description
131	F051-07	LOCALTIME	— Subclause 6.31, "<datetime value function>": The <current local time value function> — Subclause 6.30, "<datetime value expression>": When the value is a <current local time value function> — Subclause 11.5, "<default clause>": LOCALTIME option of <datetime value function>
132	F051-08	LOCALTIMESTAMP	— Subclause 6.31, "<datetime value function>": The <current local timestamp value function> — Subclause 6.30, "<datetime value expression>": When the value is a <current local timestamp value function> — Subclause 11.5, "<default clause>": LOCALTIMESTAMP option of <datetime value function>
133	**F081**	**UNION and EXCEPT in views**	— Subclause 11.22, "<view definition>": A <query expression> in a <view definition> may specify UNION, UNION ALL, and/or EXCEPT
134	**F131**	**Grouped operations**	— A *grouped view* is a view whose <query expression> contains a <group by clause>
135	F131-01	WHERE, GROUP BY, and HAVING clauses supported in queries with grouped views	— Subclause 7.4, "<table expression>": Even though a table in the <from clause> is a grouped view, the <where clause>, <group by clause>, and <having clause> may be specified
136	F131-02	Multiple tables supported in queries with grouped views	— Subclause 7.5, "<from clause>": Even though a table in the <from clause> is a grouped view, the <from clause> may specify more than one <table reference>
137	F131-03	Set functions supported in queries with grouped views	— Subclause 7.12, "<query specification>": Even though a table in the <from clause> is a grouped view, the <select list> may specify a <set function specification>
138	F131-04	Subqueries with GROUP BY and HAVING clauses and grouped views	— Subclause 7.15, "<subquery>": A <subquery> in a <comparison predicate> is allowed to contain a <group by clause> and/or a <having clause> and/or it may identify a grouped view

	Feature ID	Feature Name	Feature Description
139	F131-05	Single row SELECT with GROUP BY and HAVING clauses and grouped views	— Subclause 14.5, "<select statement: single row>": The table in a <from clause> can be a grouped view — Subclause 14.5, "<select statement: single row>": The <table expression> may specify a <group by clause> and/or a <having clause>
140	**F181**	**Multiple module support** NOTE 489 — The ability to associate multiple host compilation units with a single SQL-session at one time.	— Subclause 13.1, "<SQL-client module definition>":An SQL-agent can be associated with more than one <SQL-client module definition> With this feature, it is possible to compile <SQL-client module definition>s or <embedded SQL host program>s separately and rely on the SQL-implementation to "link" the together properly at execution time. To ensure portability, applications should adhere to the following limitations: —Avoid linking modules having cursors with the same <cursor name>. —Avoid linking modules that prepare statements using the same <SQL statement name>. —Avoid linking modules that allocate descriptors with the same <descriptor name>. —Assume that the scope of an <embedded exception declaration> is a single compilation unit. —Assume that an <embedded variable name> can be referenced only in the same compilation unit in which it is declared.
141	**F201**	**CAST function** NOTE 490 — This means the support of CAST, where relevant, among all supported data types.	— Subclause 6.12, "<cast specification>": For all supported data types — Subclause 6.25, "<value expression>": <cast specification>
142	**F221**	**Explicit defaults**	— Subclause 6.5, "<contextually typed value specification>": <default specification> NOTE 491 — Including its use in UPDATE and INSERT statements.
143	**F261**	**CASE expression**	— Subclause 6.25, "<value expression>": <case expression>
144	F261-01	Simple CASE	— Subclause 6.11, "<case expression>": The <simple case> variation
145	F261-02	Searched CASE	— Subclause 6.11, "<case expression>": The <searched case> variation

	Feature ID	Feature Name	Feature Description
146	F261-03	NULLIF	— Subclause 6.11, "<case expression>": The NULLIF <case abbreviation>
147	F261-04	COALESCE	— Subclause 6.11, "<case expression>": The COALESCE <case abbreviation>
148	**F311**	**Schema definition statement**	— Subclause 11.1, "<schema definition>"
149	F311-01	CREATE SCHEMA	— Subclause 11.1, "<schema definition>": Support for circular references in that <referential constraint definition>s in two different <table definition>s may reference columns in the other table
150	F311-02	CREATE TABLE for persistent base tables	— Subclause 11.1, "<schema definition>": A <schema element> that is a <table definition> — Subclause 11.3, "<table definition>": In the context of a <schema definition>
151	F311-03	CREATE VIEW	— Subclause 11.1, "<schema definition>": A <schema element> that is a <view definition> — Subclause 11.22, "<view definition>": In the context of a <schema definition> without the WITH CHECK OPTION clause and without support of Feature F081, "UNION and EXCEPT in views"
152	F311-04	CREATE VIEW: WITH CHECK OPTION	— Subclause 11.22, "<view definition>": The WITH CHECK OPTION clause, in the context of a <schema definition>, but without support of Feature F081, "UNION and EXCEPT in views"
153	F311-05	GRANT statement	— Subclause 11.1, "<schema definition>": A <schema element> that is a <grant statement> — Subclause 12.1, "<grant statement>": In the context of a <schema definition>
154	**F471**	**Scalar subquery values**	— Subclause 6.25, "<value expression>": A <value expression primary> can be a <scalar subquery>
155	**F481**	**Expanded NULL predicate**	— Subclause 8.7, "<null predicate>": The <row value expression> can be something other than a <column reference>

	Feature ID	Feature Name	Feature Description
156	**F812**	**Basic flagging**	— Part 1, Subclause 8.5, "SQL flagger": With "level of flagging" specified to be Core SQL Flagging and "extent of checking" specified to be Syntax Only NOTE 492 — This form of flagging identifies vendor extensions and other non-standard SQL by checking syntax only without requiring access to the catalog information.
157	**S011**	**Distinct data types**	— Subclause 11.41, "<user-defined type definition>": When <representation> is <predefined type> — Subclause 11.49, "<drop data type statement>"
158	**T321**	**Basic SQL-invoked routines**	— Subclause 11.50, "<SQL-invoked routine>" — Subclause 11.52, "<drop routine statement>" — If Feature T041, "Basic LOB data type support", is supported, then the <locator indication> clause shall also be supported NOTE 493 — "Routine" is the collective term for functions, methods, and procedures. This feature requires a conforming SQL-implementation to support both user-defined functions and user-defined procedures. An SQL-implementation that conforms to Core SQL shall support at least one language for writing routines; that language may be SQL. If the language is SQL, then the basic specification capability in Core SQL is the ability to specify a one-statement routine. Support for overloaded functions and procedures is not part of Core SQL.
159	T321-01	User-defined functions with no overloading	— Subclause 11.50, "<SQL-invoked routine>": With <function specification>
160	T321-02	User-defined stored procedures with no overloading	— Subclause 11.50, "<SQL-invoked routine>": With <SQL-invoked procedure>
161	T321-03	Function invocation	— Subclause 6.4, "<value specification> and <target specification>": With a <value expression primary> that is a <routine invocation> — Subclause 10.4, "<routine invocation>": For user-defined functions
162	T321-04	CALL statement	— Subclause 10.4, "<routine invocation>": Used by <call statement>s — Subclause 15.1, "<call statement>"
163	T321-05	RETURN statement	— Subclause 15.2, "<return statement>", if the SQL-implementation supports SQL routines

	Feature ID	Feature Name	Feature Description
164	T631	**IN predicate with one list element**	— Subclause 8.4, "<in predicate>": <in value list> containing exactly one <row value expression>

[1] A conforming SQL-implementation is required (by Clause 8, "Conformance", in ISO/IEC 9075-1) to support at least one embedded language or to support the SQL-client module binding for at least one host language.

Table 36, "Feature taxonomy for optional features", does not provide definitions of the features; the definition of those features is found in the Conformance Rules that are further summarized in Annex A, "SQL Conformance Summary".

Table 36 — Feature taxonomy for optional features

	Feature ID	Feature Name
1	**B011**	**Embedded Ada**
2	**B012**	**Embedded C**
3	**B013**	**Embedded COBOL**
4	**B014**	**Embedded Fortran**
5	**B015**	**Embedded MUMPS**
6	**B016**	**Embedded Pascal**
7	**B017**	**Embedded PL/I**
8	**B021**	**Direct** SQL
9	**B031**	**Basic dynamic SQL**
10	**B032**	**Extended dynamic SQL**
11	B032-01	<describe input statement>
12	**B033**	**Untyped SQL-invoked function arguments**
13	**B034**	**Dynamic specification of cursor attributes**
14	**B041**	**Extensions to embedded SQL exception declarations**
15	**B051**	**Enhanced execution rights**
16	**B111**	**Module language Ada**

	Feature ID	Feature Name
17	**B112**	**Module language C**
18	**B113**	**Module language COBOL**
19	**B114**	**Module language Fortran**
20	**B115**	**Module language MUMPS**
21	**B116**	**Module language Pascal**
22	**B117**	**Module language PL/I**
23	**B121**	**Routine language Ada**
24	**B122**	**Routine language C**
25	**B123**	**Routine language COBOL**
26	**B124**	**Routine language Fortran**
27	**B125**	**Routine language MUMPS**
28	**B126**	**Routine language Pascal**
29	**B127**	**Routine language PL/I**
30	**B128**	**Routine language SQL**
31	**F032**	**CASCADE drop behavior**
32	**F033**	**ALTER TABLE statement: DROP COLUMN clause**
33	**F034**	**Extended REVOKE statement**
34	F034-01	REVOKE statement performed by other than the owner of a schema object
35	F034-02	REVOKE statement: GRANT OPTION FOR clause
36	F034-03	REVOKE statement to revoke a privilege that the grantee has WITH GRANT OPTION
37	**F052**	**Intervals and datetime arithmetic**
38	**F053**	**OVERLAPS predicate**
39	**F111**	**Isolation levels other than SERIALIZABLE**
40	F111-01	READ UNCOMMITTED isolation level

	Feature ID	Feature Name
41	F111-02	READ COMMITTED isolation level
42	F111-03	REPEATABLE READ isolation level
43	**F121**	**Basic diagnostics management**
44	F121-01	GET DIAGNOSTICS statement
45	F121-02	SET TRANSACTION statement: DIAGNOSTICS SIZE clause
46	**F171**	**Multiple schemas per user**
47	**F191**	**Referential delete actions**
48	**F222**	**INSERT statement: DEFAULT VALUES clause**
49	**F231**	**Privilege tables**
50	F231-01	TABLE_PRIVILEGES view
51	F231-02	COLUMN_PRIVILEGES view
52	F231-03	USAGE_PRIVILEGES view
53	**F251**	**Domain support**
54	**F262**	**Extended CASE expression**
55	**F271**	**Compound character literals**
56	**F281**	**LIKE enhancements**
57	**F291**	**UNIQUE predicate**
58	**F301**	**CORRESPONDING in query expressions**
59	**F302**	**INTERSECT table operator**
60	F302-01	INTERSECT DISTINCT table operator
61	F302-02	INTERSECT ALL table operator
62	**F304**	**EXCEPT ALL table operator**
63	**F312**	**MERGE statement**
64	**F321**	**User authorization**
65	**F341**	**Usage tables**

	Feature ID	Feature Name
66	**F361**	**Subprogram support**
67	**F381**	**Extended schema manipulation**
68	F381-01	ALTER TABLE statement: ALTER COLUMN clause
69	F381-02	ALTER TABLE statement: ADD CONSTRAINT clause
70	F381-03	ALTER TABLE statement: DROP CONSTRAINT clause
71	**F391**	**Long identifiers**
72	**F392**	**Unicode escapes in identifiers**
73	**F393**	**Unicode escapes in literals**
74	**F401**	**Extended joined table**
75	F401-01	NATURAL JOIN
76	F401-02	FULL OUTER JOIN
77	F401-04	CROSS JOIN
78	**F402**	**Named column joins for LOBs, arrays, and multisets**
79	**F411**	**Time zone specification**
80	**F421**	**National character**
81	**F431**	**Read-only scrollable cursors**
82	F431-01	FETCH with explicit NEXT
83	F431-02	FETCH FIRST
84	F431-03	FETCH LAST
85	F431-04	FETCH PRIOR
86	F431-05	FETCH ABSOLUTE
87	F431-06	FETCH RELATIVE
88	**F441**	**Extended set function support**
89	**F442**	**Mixed column references in set functions**
90	**F451**	**Character set definition**

	Feature ID	Feature Name
91	**F461**	**Named character sets**
92	**F491**	**Constraint management**
93	**F502**	**Enhanced documentation tables**
94	F502-01	SQL_SIZING_PROFILES view
95	F502-02	SQL_IMPLEMENTATION_INFO view
96	F502-03	SQL_PACKAGES view
97	**F521**	**Assertions**
98	**F531**	**Temporary tables**
99	**F555**	**Enhanced seconds precision**
100	**F561**	**Full value expressions**
101	**F571**	**Truth value tests**
102	**F591**	**Derived tables**
103	**F611**	**Indicator data types**
104	**F641**	**Row and table constructors**
105	**F651**	**Catalog name qualifiers**
106	**F661**	**Simple tables**
107	**F671**	**Subqueries in CHECK**
108	**F672**	**Retrospective check constraints**
109	**F691**	**Collation and translation**
110	**F692**	**Enhanced collation support**
111	**F693**	**SQL-session and client module collations**
112	**F695**	**Translation support**
113	**F696**	**Additional translation documentation**
114	**F701**	**Referential update actions**
115	**F711**	**ALTER domain**

	Feature ID	Feature Name
116	F721	**Deferrable constraints**
117	F731	**INSERT column privileges**
118	F741	**Referential MATCH types**
119	F751	**View CHECK enhancements**
120	F761	**Session management**
121	F771	**Connection management**
122	F781	**Self-referencing operations**
123	F791	**Insensitive cursors**
124	F801	**Full set function**
125	F813	**Extended flagging** — Part 1, Subclause 8.5, "SQL flagger": With "level of flagging" specified to be Core SQL Flagging and "extent of checking" specified to be Catalog Lookup
126	F821	**Local table references**
127	F831	**Full cursor update**
128	F831-01	Updateable scrollable cursors
129	F831-02	Updateable ordered cursors
130	S023	**Basic structured types**
131	S024	**Enhanced structured types**
132	S025	**Final structured types**
133	S026	**Self-referencing structured types**
134	S027	**Create method by specific method name**
135	S028	**Permutable UDT options list**
136	S041	**Basic reference types**
137	S043	**Enhanced reference types**
138	S051	**Create table of type**
139	S071	**SQL paths in function and type name resolution**

	Feature ID	Feature Name
140	S081	**Subtables**
141	S091	**Basic array support**
142	S091-01	Arrays of built-in data types
143	S091-02	Arrays of distinct types
144	S091-03	Array expressions
145	S092	**Arrays of user-defined types**
146	S094	**Arrays of reference types**
147	S095	**Array constructors by query**
148	S096	**Optional array bounds**
149	S097	**Array element assignment**
150	S111	**ONLY in query expressions**
151	S151	**Type predicate**
152	S161	**Subtype treatment**
153	S162	**Subtype treatment for references**
154	S201	**SQL-invoked routines on arrays**
155	S201-01	Array parameters
156	S201-02	Array as result type of functions
157	S202	**SQL-invoked routines on multisets**
158	S211	**User-defined cast functions**
159	S231	**Structured type locators**
160	S232	**Array locators**
161	S233	**Multiset locators**
162	S241	**Transform functions**
163	S242	**Alter transform statement**
164	S251	**User-defined orderings**

	Feature ID	Feature Name
165	S261	**Specific type method**
166	S271	**Basic multiset support**
167	S272	**Multisets of user-defined types**
168	S274	**Multisets of reference types**
169	S275	**Advanced multiset support**
170	S281	**Nested collection types**
171	S291	**Unique constraint on entire row**
172	T011	**Timestamp in Information Schema**
173	T031	**BOOLEAN data type**
174	T041	**Basic LOB data type support**
175	T041-01	BLOB data type — Subclause 5.2, "<token> and <separator>": The <reserved word>s BINARY, BLOB, LARGE, and OBJECT — Subclause 5.3, "<literal>": <binary string literal> — Subclause 6.1, "<data type>": The BINARY LARGE OBJECT data type — Subclause 6.28, "<string value expression>": For values of type BINARY LARGE OBJECT — Subclause 13.6, "Data type correspondences": Type correspondences for BINARY LARGE OBJECT for all supported languages
176	T041-02	CLOB data type — Subclause 5.2, "<token> and <separator>": The <reserved word>s CHARACTER, CLOB, LARGE, and OBJECT — Subclause 6.1, "<data type>": The CHARACTER LARGE OBJECT data type — Subclause 6.28, "<string value expression>": For values of type CHARACTER LARGE OBJECT — Subclause 13.6, "Data type correspondences": Type correspondences for CHARACTER LARGE OBJECT for all supported languages — The implicit casting among the fixed-length and variable-length character string types supported by subfeature E021-10 is extended to support the character large object type

	Feature ID	Feature Name
177	T041-03	POSITION, LENGTH, LOWER, TRIM, UPPER, and SUBSTRING functions for LOB data types — Subclause 6.27, "<numeric value function>": The <position expression> for expressions of type BINARY LARGE OBJECT and CHARACTER LARGE OBJECT — Subclause 6.27, "<numeric value function>": The <char length expression> for expressions of type CHARACTER LARGE OBJECT — Subclause 6.27, "<numeric value function>": The <octet length expression> for expressions of type BINARY LARGE OBJECT and CHARACTER LARGE OBJECT — Subclause 6.29, "<string value function>": The <fold> function for expressions of type CHARACTER LARGE OBJECT — Subclause 6.29, "<string value function>": The <trim function> for expressions of type CHARACTER LARGE OBJECT — Subclause 6.29, "<string value function>": The <blob trim function> — Subclause 6.29, "<string value function>": The <character substring function> for expressions of type CHARACTER LARGE OBJECT — Subclause 6.29, "<string value function>": The <blob substring function>
178	T041-04	Concatenation of LOB data types — Subclause 6.28, "<string value expression>": The <concatenation> expression for expressions of type CHARACTER LARGE OBJECT — Subclause 6.28, "<string value expression>": The <blob concatenation> expression
179	T041-05	LOB locator: non-holdable — Subclause 13.3, "<externally-invoked procedure>": <locator indication> — Subclause 14.14, "<free locator statement>"
180	**T042**	**Extended LOB data type support**
181	**T051**	**Row types**
182	**T052**	**MAX and MIN for row types**
183	**T053**	**Explicit aliases for all-fields reference**
184	**T061**	**UCS support**
185	**T071**	**BIGINT data type**
186	**T111**	**Updatable joins, unions, and columns**
187	**T121**	**WITH (excluding RECURSIVE) in query expression**
188	**T122**	**WITH (excluding RECURSIVE) in subquery**
189	**T131**	**Recursive query**
190	**T132**	**Recursive query in subquery**
191	**T141**	**SIMILAR predicate**
192	**T151**	**DISTINCT predicate**

	Feature ID	Feature Name
193	T152	**DISTINCT predicate with negation**
194	T171	**LIKE clause in table definition**
195	T172	**AS subquery clause in table definition**
196	T173	**Extended LIKE clause in table definition**
197	T174	**Identity columns**
198	T175	**Generated columns**
199	T176	**Sequence generator support**
200	T191	**Referential action RESTRICT**
201	T201	**Comparable data types for referential constraints**
202	T211	**Basic trigger capability**
203	T211-01	Triggers activated on UPDATE, INSERT, or DELETE of one base table.
204	T211-02	BEFORE triggers
205	T211-03	AFTER triggers
206	T211-04	FOR EACH ROW triggers
207	T211-05	Ability to specify a search condition that shall be *True* before the trigger is invoked.
208	T211-06	Support for run-time rules for the interaction of triggers and constraints.
209	T211-07	TRIGGER privilege
210	T211-08	Multiple triggers for the same event are executed in the order in which they were created in the catalog.
211	T212	**Enhanced trigger capability**
212	T231	**Sensitive cursors**
213	T241	**START TRANSACTION statement**
214	T251	**SET TRANSACTION statement: LOCAL option**
215	T261	**Chained transactions**
216	T271	**Savepoints**

	Feature ID	Feature Name
217	T272	Enhanced savepoint management
218	T281	SELECT privilege with column granularity
219	T301	Functional dependencies
220	T312	OVERLAY function
221	T322	Overloading of SQL-invoked functions and procedures
222	T323	Explicit security for external routines
223	T324	Explicit security for SQL routines
224	T325	Qualified SQL parameter references
225	T326	Table functions
226	T331	Basic roles
227	T332	Extended roles
228	T351	Bracketed SQL comments (/*...*/ comments)
229	T431	Extended grouping capabilities
230	T432	Nested and concatenated GROUPING SETS
231	T433	Multiargument GROUPING function
232	T434	GROUP BY DISINCT
233	T441	ABS and MOD functions
234	T461	Symmetric BETWEEN predicate
235	T471	Result sets return value
236	T491	LATERAL derived table
237	T501	Enhanced EXISTS predicate
238	T511	Transaction counts
239	T551	Optional key words for default syntax
240	T561	Holdable locators
241	T571	Array-returning external SQL-invoked functions

	Feature ID	Feature Name
242	T572	Multiset-returning external SQL-invoked functions
243	T581	Regular expression substring function
244	T591	UNIQUE constraints of possibly null columns
245	T601	Local cursor references
246	T611	Elementary OLAP operations
247	T612	Advanced OLAP operations
248	T613	Sampling
249	T621	Enhanced numeric functions
250	T641	Multiple column assignment
251	T651	SQL-schema statements in SQL routines
252	T652	SQL-dynamic statements in SQL routines
253	T653	SQL-schema statements in external routines
254	T654	SQL-dynamic statements in external routines
255	T655	Cyclically dependent routines

This page intentionally left blank.

Annex G

(informative)

Defect Reports not addressed in this edition of ISO/IEC 9075

Each entry in this Annex describes a reported defect in the previous edition of this part of ISO/IEC 9075 that remains in this edition.

1) **Subclause 10.4, "<routine invocation>":**

   There is no definition of how to pass values of type BOOLEAN or of large object types as arguments to invocations of external routines. More generally, the question of how to convert a value of any SQL type to a value of an appropriate host language type at the interface to an SQL-invoked routine is not addressed. The rules in Subclause 13.4, "Calls to an <externally-invoked procedure>", are appropriate, but they are not referenced by the rules of Subclause 10.4, "<routine invocation>".

2) **Subclause 20.1, "<embedded SQL host program>":**

   SR 21)h)i)6) and SR 21)l)i)3)B)VI) both refer to the SQL data type that corresponds to a given host language data type, as determined by application of the rules in Subclause 13.6, "Data type correspondences". These two syntax rules are sometimes ambiguous, because Subclause 13.6, "Data type correspondences" does not always give exactly one SQL data type for a given host language type, as can be seen by inspection of the data type correspondence tables given in that Subclause. For example, Table 17, "Data type correspondences for C", in which the C data type "pointer to long" maps to both INTEGER and BOOLEAN.

3) **Subclause 16.3, "<set constraints mode statement>":**

   There are several problems with the deferred constraint checking specified by use of the keyword DEFERRED:

   a) Exactly when <referential action>s of deferred referential constraints are processed is not precisely specified.

   b) When referential constraint checking is immediate and execution of an SQL-statement causes rows to be deleted, those rows are merely "marked for deletion" and not actually deleted until all constraint checking has been done. This ensures the correct processing of <referential action>s, such as ON DELETE SET DEFAULT, that cause changes to SQL-data. When the checking of some referential constraint has been deferred and the mode of that constraint is set to IMMEDIATE, it can happen that a row whose presence is needed to ensure the correct processing of a <referential action> has been actually deleted, as a result of the successful prior execution of some SQL-data change statement.

      For example, consider:

```
CREATE TABLE T1 ( A INTEGER, PRIMARY KEY ( A ) ) ;

CREATE TABLE T2 ( A INTEGER,
  CONSTRAINT C1
    FOREIGN KEY ( A ) REFERENCES T2
```

```
    ON UPDATE CASCADE
    DEFERRABLE) ;

INSERT INTO T1 VALUES ( 1 ) ;

INSERT INTO T2 VALUES ( 1 ) ;
```

Now VALUES (1) is a matching row in T2 for the only row in T1.

```
SET CONSTRAINTS C1 DEFERRED ;
UPDATE T1 SET A = 9 ;
```

The processing of the UPDATE statement causes the only row in T1 to change from VALUES (1) to VALUES (9). The update to that row in T1 is supposed to be propagated to the only row in T2 under the rule for constraint C1, but processing of constraint C1 is deferred and does not take place at this time.

```
SET CONSTRAINTS C1 IMMEDIATE ;
```

Constraint C1 is processed now, but, there now being no row in T1 for which the VALUES (1) in T2 is a matching row, no change to T2 will take place under the General Rules of Subclause 11.8, "<referential constraint definition>". Thus, the constraint is violated in spite of the existence of the <referential action> that is expected to prevent this from happening. Moreover, the unmatched row in T2 remains even after the mode of C1 has been set to IMMEDIATE, for a <set constraints mode statement> does not make any changes to SQL-data that might be canceled under the General Rules of Subclause 13.5, "<SQL procedure statement>".

c) Various problems have been noted with the possible interaction of deferred processing of referential constraints and triggers. For example, consider:

```
CREATE TABLE T1 ( A INTEGER, PRIMARY KEY ( A ) ) ;

CREATE TABLE T2 (A INTEGER,
  PRIMARY KEY ( A ),
  CONSTRAINT C1
    FOREIGN KEY ( A ) REFERENCES T1
      ON DELETE CASCADE
      DEFERRABLE ) ;

CREATE TABLE T3 ( A INTEGER, PRIMARY KEY ( A ) ) ;

CREATE TRIGGER TR1 AFTER DELETE ON T2
  FOR EACH STATEMENT
    INSERT INTO T3 VALUES ( ( SELECT COUNT(*) FROM T3 ) + 1 ) ;

INSERT INTO T1 VALUES ( 1 ), ( 2 ) ;

INSERT INTO T2 VALUES ( 1 ), ( 2 ) ;

DELETE FROM T1 WHERE A = 1 ;
```

Because constraint checking is immediate, this deletion from T1 will cause the same row, VALUES (1), to be deleted from T2, thus activating trigger TR1 and causing VALUES (1) to be inserted into T3.

```
DELETE FROM T1 WHERE A = 2 ;
```

Likewise, this deletion will cause the row VALUES (2) to be deleted from T2, thus activating trigger TR1 and causing VALUES (1) to be inserted into T3..

Likewise, this deletion will cause the row VALUES (2) to be deleted from T2, thus activating trigger TR1 for a second time and causing VALUES (2) to be inserted into T3.

But if SET CONSTRAINTS C1 DEFERRED is executed immediately before those two deletions, then the deletions from T1 will not be propagated to T2 and so trigger TR1 will not be activated and T3 will not be updated. When SET CONSTRAINTS C1 IMMEDIATE is subsequently executed, even if the problem illustrated in point b) is addressed so that the deletions are somehow now propagated to T2 after all, it is not clear how many activations of trigger TR1 will take place (nor, if there are more than one, in what order those activations will take place).

This page intentionally left blank.

Index

Index entries appearing in **boldface** indicate the page where the word, phrase, or BNF nonterminal was defined; index entries appearing in *italics* indicate a page where the BNF nonterminal was used in a Format; and index entries appearing in roman type indicate a page where the word, phrase, or BNF nonterminal was used in a heading, Function, Syntax Rule, Access Rule, General Rule, Leveling Rule, Table, or other descriptive text.

— A —

— B —

— C —

— D —

— E —

— F —

— G —

— H —

— I —

— J —

— K —

— L —

— M —

— N —

— O —

— P —

— Q —

— R —

— S —

— T —

— U —

— V —

— W —

— X —

— Y —

— Z —

Proposition de modification

N'hésitez pas à nous faire part de vos suggestions et de vos commentaires. Au moment de soumettre des propositions de modification aux normes CSA et autres publications CSA prière de fournir les renseignements demandés ci-dessous et de formuler les propositions sur une feuille volante. Il est recommandé d'inclure

- le numéro de la norme/publication
- le numéro de l'article, du tableau ou de la figure visé
- la formulation proposée
- la raison de cette modification.

Proposal for change

CSA welcomes your suggestions and comments. To submit your proposals for changes to CSA Standards and other CSA publications, please supply the information requested below and attach your proposal for change on a separate page(s). Be sure to include the

- Standard/publication number
- relevant Clause, Table, and/or Figure number(s)
- wording of the proposed change
- rationale for the change.

Nom/Name: _____

Affiliation: _____

Adresse/Address: _____

_____

Ville/City: _____

État/Province/State: _____

Pays/Country: _____ **Code postal/Postal/Zip code:** _____

Téléphone/Telephone: _____ **Télécopieur/Fax:** _____

Date: _____

J'accepte que la CSA conserve et utilise les renseignements ci-dessus afin de faciliter la réception de mes suggestions et commentaires.

I consent to CSA collecting and using the above information to facilitate the collection of my suggestions and comments.

Consultez la politique CSA en matière de confidentialité au www.csagroup.org/legal pour savoir comment nous protégeons vos renseignements personnels.

Visit CSA's policy on privacy at www.csagroup.org/legal to find out how we protect your personal information.

ICS 35.060