ISO/IEC JTC 1/SC 32 N 0292

Date: 1999-07-07

REPLACES: --

ISO/IEC JTC 1/SC 32

Data Management and Interchange

Secretariat: United States of America (ANSI)

Administered by Pacific Northwest National Laboratory on behalf of ANSI

DOCUMENT TYPE	Working Draft Amendment (for information or comment)
TITLE	ISO/IEC WD 9075 Amd 1 ISO-ANSI Working Draft —Database Language SQL — Amendment 1: On-Line Analytical Processing (SQL/OLAP) «Amendment 1»
SOURCE	Jim Melton, Editor
PROJECT NUMBER	03.04.01.01.00
STATUS	For review and comment into Working Group 2
REFERENCES	
ACTION ID.	FYI
REQUESTED ACTION	Participation in WG 2 editing is requested.
DUE DATE	
Number of Pages	83
LANGUAGE USED	English
DISTRIBUTION	P & L Members SC Chair WG Conveners and Secretaries

Douglas Mann, Secretariat, ISO/IEC JTC 1/SC 32

Pacific Northwest National Laboratory *, 901 D Street, SW., Suite 900, Washington, DC, 20024-2115, United States of America

Telephone: +1 703 575 2114; Facsimile; +1 703 681 9180; E-mail: MannD@battelle.org

*Pacific Northwest National Laboratory (PNL) administers the ISO/IEC JTC 1/SC 32 Secretariat on behalf of ANSI

ISO-ANSI Working Draft — Database Language SQL — Amendment 1: On-Line Analytical Processing (SQL/OLAP) «Amendment 1»

June 1999

Contents

Page

TABLES

Tables **Page**

Foreword

ANSI Only—caused by ISO changes not yet considered by ANSI

To be supplied

ISO Only—caused by ANSI changes not yet considered by ISO

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75% of the national bodies casting a vote.

International Standard ISO/IEC 9075, was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 32 , *Data management and interchange*.

ISO/IEC 9075 consists of the following parts, under the general title *Information technology — Database languages — SQL*:

— *Part 1: Framework (SQL/Framework)*

— *Part 2: Foundation (SQL/Foundation)*

— *Part 3: Call-Level Interface (SQL/CLI)*

— *Part 4: Persistent Stored Modules (SQL/PSM)*

— *Part 5: Host Language Bindings (SQL/Bindings)*

Annexes A, B, C, and D of this amendment to ISO/IEC 9075 are for information only.

Introduction

The organization of this amendment to
[ANSI] ANSI X3.135
[ISO] ISO/IEC 9075
is as follows:

1) Clause 1, "Scope", specifies the scope of this amendment to
   [ANSI] ANSI X3.135.
   [ISO] ISO/IEC 9075.

2) Clause 2, "Normative references", identifies additional standards that, through reference in this amendment to ISO/IEC 9075, constitute provisions of this amendment to
   [ANSI] ANSI X3.135.
   [ISO] ISO/IEC 9075.

3) Clause 3, "Definitions, notations, and conventions", defines the notations and conventions used in this amendment to
   [ANSI] ANSI X3.135.
   [ISO] ISO/IEC 9075.

4) Clause 4, "Concepts", presents concepts used in the definition of OLAP facilities.

5) Clause 5, "Lexical elements", defines a number of lexical elements used in the definition of OLAP facilities.

6) Clause 6, "Scalar expressions", defines a number of scalar expressions used in the definition of OLAP facilities.

7) Clause 7, "Query expressions", defines the elements of the language that produce rows and tables of data as used in OLAP facilities.

8) Clause 8, "Additional common elements", defines additional common elements used in the definition of OLAP facilities.

9) Clause 9, "Schema definition and manipulation", defines the schema definition and manipulation statements associated with the definition of OLAP facilities.

10) Clause 10, "SQL-client modules", defines SQL-client modules and externally-invoked procedures.

11) Clause 11, "Data manipulation", defines data manipulation operations associated with OLAP facilities.

12) Clause 12, "Dynamic SQL", defines the SQL dynamic statements.

13) Clause 13, "Status codes", defines SQLSTATE values related to OLAP facilities.

14) Clause 14, "Conformance", defines the criteria for conformance to this amendment to
ANSI ANSI X3.135.
ISO ISO/IEC 9075.

15) Annex A, "SQL conformance summary", is an informative Annex. It summarizes the conformance requirements of the SQL language.

16) Annex B, "Implementation-defined elements", is an informative Annex. It lists those features for which the body of this amendment to
ANSI ANSI X3.135
ISO ISO/IEC 9075
states that the syntax, the meaning, the returned results, the effect on SQL-data and/or schemas, or any other behavior is partly or wholly implementation-defined.

17) Annex C, "Implementation-dependent elements", is an informative Annex. It lists those features for which the body of this amendment to
ANSI ANSI X3.135
ISO ISO/IEC 9075
states that the syntax, the meaning, the returned results, the effect on SQL-data and/or schemas, or any other behavior is partly or wholly implementation-dependent.

18) Annex D, "SQL feature and package taxonomy", is an informative Annex. It identifies features of the SQL language specified in this amendment to
ANSI ANSI X3.135
ISO ISO/IEC 9075
by a numeric identifier and a short descriptive name. This taxonomy is used to specify conformance to Core SQL and may be used to develop other profiles involving the SQL language.

In the text of this amendment to
ANSI ANSI X3.135,
ISO ISO/IEC 9075,
Clauses begin a new odd-numbered page, and in Clause 5, "Lexical elements", through Clause 14, "Conformance", Subclauses begin a new page. Any resulting blank space is not significant.

Information technology — Database languages — SQL —

Amendment 1:

On-Line Analytical Processing (SQL/OLAP)

1 Scope

This amendment to
ANSI American Standard ANSI X3.135
ISO International Standard ISO/IEC 9075
specifies the syntax and semantics of database language facilities that support on-line analytical processing.

The database language facilities that support on-line analytical processing include:

— To Be Supplied

NOTE 1 – The context for this amendment to
ANSI ANSI X3.135
ISO ISO/IEC 9075
is described by the Reference Model of Data Management (ISO/IEC 10032:1993).

2 Normative references

The following standards contain provisions that, through reference in this text, constitute provisions of this amendment to
[ANSI] ANSI X3.135.
[ISO] ISO/IEC 9075.
At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this amendment to
[ANSI] ANSI X3.135
[ISO] ISO/IEC 9075
are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

ANSI Only—caused by ISO changes not yet considered by ANSI

ANSI/ISO/IEC 9075-1:1999, *Information technology — Database languages — SQL — Part 1: Framework (SQL/Framework).*

ANSI/ISO/IEC 9075-2:1999, *Information technology — Database languages — SQL — Part 2: Foundation (SQL/Foundation).*

ANSI/ISO/IEC 9075-3:1999, *Information technology — Database languages — SQL — Part 3: Call-Level Interface (SQL/CLI).*

ANSI/ISO/IEC 9075-4:1999, *Information technology — Database languages — SQL — Part 4: Persistent Stored Modules (SQL/PSM).*

ANSI/ISO/IEC 9075-5:1999, *Information technology — Database languages — SQL — Part 5: Host Language Bindings (SQL/Bindings).*

ISO Only—caused by ANSI changes not yet considered by ISO

ISO/IEC 9075-1:1999, *Information technology — Database languages — SQL — Part 1: Framework (SQL/Framework).*

ISO/IEC 9075-2:1999, *Information technology — Database languages — SQL — Part 2: Foundation (SQL/Foundation).*

ISO/IEC 9075-3:1999, *Information technology — Database languages — SQL — Part 3: Call-Level Interface (SQL/CLI).*

ISO/IEC 9075-4:1999, *Information technology — Database languages — SQL — Part 4: Persistent Stored Modules (SQL/PSM).*

ISO/IEC 9075-5:1999, *Information technology — Database languages — SQL — Part 5: Host Language Bindings (SQL/Bindings).*

---

3 Definitions, notations, and conventions

3.1 Definitions

Insert this paragraph For the purposes of this amendment to
ANSI ANSI X3.135,
ISO ISO/IEC 9075,
the definitions given in
ANSI ANSI X3.135.1 and ANSI X3.135.2
ISO ISO/IEC 9075-1 and ISO/IEC 9075-2
apply.

3.1.1 Definitions provided in Amendment 1

To Be Supplied

3.2 Notations

Insert this paragraph The syntax notation used in this amendment to
ANSI ANSI X3.135
ISO ISO/IEC 9075
is an extended version of BNF ("Backus Normal Form" or "Backus Naur Form"). This version of BNF is fully described in Subclause 7.1, "Notation", of
ANSI ANSI X3.135.1.
ISO ISO/IEC 9075-1.

3.3 Conventions

Insert this paragraph Except as otherwise specified in this amendment to
ANSI ANSI X3.135,
ISO ISO/IEC 9075,
the conventions used in this amendment to
ANSI ANSI X3.135
ISO ISO/IEC 9075
are identical to those described in
ANSI ANSI X3.135.1 and ANSI X3.135.2.
ISO ISO/IEC 9075-1 and ISO/IEC 9075-2.

ANSI Only—caused by ISO changes not yet considered by ANSI

3.3.1 Relationships to other parts of ANSI X3.135

ISO Only—caused by ANSI changes not yet considered by ISO

3.3.2 Relationships to other parts of ISO/IEC 9075

3.3.2.1 Clause, Subclause, and Table relationships

Table 1—Clause, Subclause, and Table relationships

ANSI Only—caused by ISO changes not yet considered by ANSI

ISO Only—caused by ANSI changes not yet considered by ISO

Clause, Subclause, or Table in this part of ANSI X3.135 Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part Corresponding Clause, Subclause, or Table from another part	Part containing correspondence Part containing correspondence
Clause 1, "Scope"	Clause 1, "Scope"	ANSI ANSI X3.135.2 ISO ISO/IEC 9075-2
Clause 2, "Normative references"	Clause 2, "Normative references"	ANSI ANSI X3.135.2 ISO ISO/IEC 9075-2
Clause 3, "Definitions, notations, and conventions"	Clause 3, "Definitions, notations, and conventions"	ANSI ANSI X3.135.2 ISO ISO/IEC 9075-2
Subclause 3.1, "Definitions"	Subclause 3.1, "Definitions"	ANSI ANSI X3.135.2 ISO ISO/IEC 9075-2

Table 1—Clause, Subclause, and Table relationships (Cont.)

Clause, Subclause, or Table in this part of ANSI X3.135 **Clause, Subclause, or Table in this part of ISO/IEC 9075**	**Corresponding Clause, Subclause, or Table from another part** **Corresponding Clause, Subclause, or Table from another part**	**Part containing correspondence** **Part containing correspondence**
Subclause 3.1.1, "Definitions provided in Amendment 1"	*(none)*	*(none)*
Subclause 3.3, "Conventions"	Subclause 3.3, "Conventions"	ANSI ANSI X3.135.2 ISO ISO/IEC 9075-2
Subclause 3.3.2, "Relationships to other parts of ISO/IEC 9075"	*(none)*	*(none)*
Subclause 3.3.2.1, "Clause, Subclause, and Table relationships"	*(none)*	*(none)*
Clause 4, "Concepts"	Clause 4, "Concepts"	ANSI ANSI X3.135.2 ISO ISO/IEC 9075-2
Subclause 4.1, "Query functions"	*(none)*	*(none)*
Subclause 4.1.1, "Grouped table functions"	*(none)*	*(none)*
Subclause 4.1.2, "OLAP functions"	*(none)*	*(none)*
Subclause 4.1.3, "Tables"	Subclause 4.16, "Tables"	ANSI ANSI X3.135.2 ISO ISO/IEC 9075-2
Subclause 4.1.3.1, "Windows"	*(none)*	*(none)*
Subclause 4.2, "SQL-invoked routines"	Subclause 4.23, "SQL-invoked routines"	ANSI ANSI X3.135.2 ISO ISO/IEC 9075-2
Clause 5, "Lexical elements"	Clause 5, "Lexical elements"	ANSI ANSI X3.135.2 ISO ISO/IEC 9075-2
Subclause 5.1, "<token> and <separator>"	Subclause 5.2, "<token> and <separator>"	ANSI ANSI X3.135.2 ISO ISO/IEC 9075-2
Subclause 5.1.1, "Names and identifiers"	Subclause 5.4, "Names and identifiers"	ANSI ANSI X3.135.2 ISO ISO/IEC 9075-2

Table 1—Clause, Subclause, and Table relationships (Cont.)

Clause, Subclause, or Table in this part of ANSI X3.135 Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part Corresponding Clause, Subclause, or Table from another part	Part containing correspondence Part containing correspondence
Clause 6, "Scalar expressions"	Clause 6, "Scalar expressions"	ANSI ANSI X3.135.2 ISO ISO/IEC 9075-2
Subclause 6.1, "<set function specification>"	Subclause 6.16, "<set function specification>"	ANSI ANSI X3.135.2 ISO ISO/IEC 9075-2
Subclause 6.2, "<OLAP function>"	*(none)*	*(none)*
Subclause 6.3, "<value expression>"	Subclause 6.23, "<value expression>"	ANSI ANSI X3.135.2 ISO ISO/IEC 9075-2
Clause 7, "Query expressions"	Clause 7, "Query expressions"	ANSI ANSI X3.135.2 ISO ISO/IEC 9075-2
Subclause 7.1, "<table expression>"	Subclause 7.4, "<table expression>"	ANSI ANSI X3.135.2 ISO ISO/IEC 9075-2
Subclause 7.2, "<joined table>"	Subclause 7.7, "<joined table>"	ANSI ANSI X3.135.2 ISO ISO/IEC 9075-2
Subclause 7.3, "<where clause>"	Subclause 7.8, "<where clause>"	ANSI ANSI X3.135.2 ISO ISO/IEC 9075-2
Subclause 7.4, "<having clause>"	Subclause 7.10, "<having clause>"	ANSI ANSI X3.135.2 ISO ISO/IEC 9075-2
Subclause 7.5, "<window clause>"	*(none)*	*(none)*
Subclause 7.6, "<query specification>"	Subclause 7.11, "<query specification>"	ANSI ANSI X3.135.2 ISO ISO/IEC 9075-2

Table 1—Clause, Subclause, and Table relationships (Cont.)

Clause, Subclause, or Table in this part of ANSI X3.135 **Clause, Subclause, or Table in this part of ISO/IEC 9075**	**Corresponding Clause, Subclause, or Table from another part** **Corresponding Clause, Subclause, or Table from another part**	**Part containing correspondence** **Part containing correspondence**
Clause 8, "Additional common elements"	Clause 10, "Additional common elements"	ANSI ANSI X3.135.2 ISO ISO/IEC 9075-2
Subclause 8.1, "<aggregate function>"	*(none)*	*(none)*
Subclause 8.2, "<sort specification list>"	*(none)*	*(none)*
Clause 9, "Schema definition and manipulation"	Clause 11, "Schema definition and manipulation"	ANSI ANSI X3.135.2 ISO ISO/IEC 9075-2
Subclause 9.1, "<drop routine statement>"	Subclause 11.51, "<drop routine statement>"	ANSI ANSI X3.135.2 ISO ISO/IEC 9075-2
Subclause 9.2, "<drop user-defined ordering statement>"	Subclause 11.55, "<drop user-defined ordering statement>"	ANSI ANSI X3.135.2 ISO ISO/IEC 9075-2
Clause 10, "SQL-client modules"	Subclause 4.21, "SQL-client modules"	ANSI ANSI X3.135.2 ISO ISO/IEC 9075-2
Subclause 10.1, "Calls to an <externally-invoked procedure>"	Subclause 13.4, "Calls to an <externally-invoked procedure>"	ANSI ANSI X3.135.2 ISO ISO/IEC 9075-2
Clause 11, "Data manipulation"	Clause 14, "Data manipulation"	ANSI ANSI X3.135.2 ISO ISO/IEC 9075-2
Subclause 11.1, "<declare cursor>"	Subclause 14.1, "<declare cursor>"	ANSI ANSI X3.135.2 ISO ISO/IEC 9075-2
Subclause 11.2, "<select statement: single row>"	Subclause 14.5, "<select statement: single row>"	ANSI ANSI X3.135.2 ISO ISO/IEC 9075-2

Table 1—Clause, Subclause, and Table relationships (Cont.)

Clause, Subclause, or Table in this part of ANSI X3.135 Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part Corresponding Clause, Subclause, or Table from another part	Part containing correspondence Part containing correspondence
Clause 12, "Dynamic SQL"	Clause 15, "Dynamic SQL"	ANSI ANSI X3.135.5 ISO ISO/IEC 9075-5
Subclause 12.1, "<prepare statement>"	Subclause 15.6, "<prepare statement>"	ANSI ANSI X3.135.5 ISO ISO/IEC 9075-5
Clause 13, "Status codes"	Clause 22, "Status codes"	ANSI ANSI X3.135.2 ISO ISO/IEC 9075-2
Subclause 13.1, "SQLSTATE"	Subclause 22.1, "SQLSTATE"	ANSI ANSI X3.135.2 ISO ISO/IEC 9075-2
Clause 14, "Conformance"	Clause 23, "Conformance"	ANSI ANSI X3.135.2 ISO ISO/IEC 9075-2
Subclause 14.1, "General conformance requirements"	Subclause 23.1, "General conformance requirements"	ANSI ANSI X3.135.2 ISO ISO/IEC 9075-2
Annex A, "SQL conformance summary"	Appendix A, "SQL Conformance Summary"	ANSI ANSI X3.135.2 ISO ISO/IEC 9075-2
Annex B, "Implementation-defined elements"	Appendix B, "Implementation-defined elements"	ANSI ANSI X3.135.2 ISO ISO/IEC 9075-2
Annex C, "Implementation-dependent elements"	Appendix C, "Implementation-dependent elements"	ANSI ANSI X3.135.2 ISO ISO/IEC 9075-2
Annex D, "SQL feature and package taxonomy"	Appendix F, "SQL feature and package taxonomy"	ANSI ANSI X3.135.2 ISO ISO/IEC 9075-2

Table 1—Clause, Subclause, and Table relationships (Cont.)

Clause, Subclause, or Table in this part of ANSI X3.135 **Clause, Subclause, or Table in this part of ISO/IEC 9075**	**Corresponding Clause, Subclause, or Table from another part** **Corresponding Clause, Subclause, or Table from another part**	**Part containing correspondence** **Part containing correspondence**
Table 1, "Clause, Subclause, and Table relationships"	*(none)*	*(none)*
Table 2, "SQLSTATE class and subclass values"	Table 27, "SQLSTATE class and subclass values"	ANSI ANSI X3.135.2 ISO ISO/IEC 9075-2
Table 3, "Implied feature relationships"	Table 30, "Implied feature relationships"	ANSI ANSI X3.135.2 ISO ISO/IEC 9075-2
Table 4, "SQL/OLAP feature taxonomy for features outside Core SQL"	Table 32, "SQL/Foundation feature taxonomy for features outside Core SQL"	ANSI ANSI X3.135.2 ISO ISO/IEC 9075-2

4 Concepts

4.1 Query functions

Query functions are functions that may only be invoked as part of a <query specification> or <select statement: single row>, and then only in certain contexts, identified below. There are two kinds of query functions:

— Grouped table functions, which compute a grouping operation or an aggregate function from a group of a grouped table.

— OLAP functions, which compute a rank, row number, or window aggregate function.

Grouped table functions may appear only in the <select list>, <having clause>, or <window clause> of a <query specification> or <select statement: single row>, or in the <order by clause> of a cursor that is a simple table query. OLAP functions may only appear in the <select list> of a <query specification> or <select statement: single row>, or in the <order by clause> of a simple table query.

4.1.1 Grouped table functions

There are two kinds of grouped table functions: the *grouping operation* and *grouped table aggregate functions*.

The only grouping operation is `GROUPING(<column reference>)`, which returns a value of 1 (one) to indicate a row that aggregates over the <column reference> during the execution of a grouped query containing CUBE and ROLLUP, and returns 0 (zero) otherwise.

One grouped table aggregate function is `COUNT (*)`, which computes the number of rows in a group of a grouped query.

All other grouped table aggregate functions take an arbitrary <value expression> as the argument, optionally qualified with either DISTINCT or ALL. If DISTINCT is specified, then redundant duplicates are removed from the group before performing the calculation. In addition, all rows in which the <value expression> evaluates to null are removed from the group. If there are no rows remaining in the group, the COUNT is 0 (zero), and any other grouped table aggregate function is null. Otherwise, the computation concludes and the result is:

— If COUNT with a <value expression> argument is specified, then the number of rows remaining in the group.

— If SUM is specified, then the sum of <value expression> evaluated for each row remaining in the group.

— If AVG is specified, then the average of <value expression> evaluated for each row remaining in the group.

— If MAX is specified, then the maximum value of <value expression> evaluated for each row remaining in the group.

— If MIN is specified, then the maximum value of <value expression> evaluated for each row remaining in the group.

— If EVERY is specified, then *true* if the <value expression> evaluates to *true* for every row remaining in the group; otherwise, *false* .

— If ANY or SOME is specified, then *true* if the <value expression> evaluates to *true* for at least one row remaining in the group; otherwise, *false* .

4.1.2 OLAP functions

There are three kinds of OLAP functions: *rank functions*, *row number function*, and *window aggregate functions*.

The rank functions compute the ordinal rank of a row *R* within the partition of *R* as defined by a window descriptor, according to the ordering of those rows, also specified by the same window descriptor. Rows that are not distinct with respect to the ordering within their partition are assigned the same rank. There are two variants, indicated by the keywords RANK and DENSERANK.

— If RANK is specified, then the rank of row *R* is defined as 1 (one) plus the number of rows that precede *R* and are not peers of *R*.

NOTE 2 – This implies that if two or more rows are not distinct with respect to the ordering, then there will be one or more gaps in the sequential rank numbering.

— If DENSERANK is specified, then the rank of row *R* is defined as the number of rows preceding and including *R* that are distinct with respect to the ordering.

NOTE 3 – This implies that there are no gaps in the sequential rank numbering of rows in each partition.

The ROWNUMBER function computes the sequential row number, starting with 1 (one) for the first row, of the row within its partition according to the ordering of the window.

The window aggregate functions compute an aggregate value (COUNT, SUM, AVG, *etc.*), the same as a grouped table aggregate function, except that the computation aggregates over the aggregation group of a row rather than over a group of a grouped table.

4.1.3 Tables

4.1.3.1 Windows

A *window* is a temporary data structure associated with a <table expression>. A window is defined explicitly by a <window definition> or implicitly by an <inline window specification>. Implicitly defined windows have an implementation-dependent window name. A window is used to specify partitions and aggregation groups, which are multisets of rows used in the definition of <OLAP function>s.

Every window defines a partitioning of the rows of the <table expression>. The partitioning is specified by a list of columns. Partitioning is similar to forming groups of a grouped table. However, unlike grouped tables, each row is retained in the result of the <table expression>. The partition of a row *R* is the multiset of rows *R2* that are not distinct from *R*, for all columns enumerated in the <window partition clause>. Partitioning is optional; if omitted, there is a single partition consisting of all the rows in the result.

NOTE 4 – If a <table expression> is grouped and also has a window, there is a syntactic transformation that segregates the grouping into a <derived table>, so that the window partitions rows of the <derived table> rather than groups of rows.

A window may define an ordering of rows within each partition defined by the window. The ordering of rows within partitions is specified by a list of <value expression>s, followed by ASC (for ascending order) or DESC (for descending order). In addition, NULLS FIRST or NULLS LAST may be specified, to indicate whether a null value should appear before or after all non-null values in the ordered sequence of each <value expression>.

Optionally, a window may define an aggregation group for each row *R*. An aggregation group is always defined relative to the current row. An aggregation group is specified by three syntactic elements:

— The choice of RANGE, to indicate a logical definition of the aggregation group by offsetting forward or backward from the current row by an increment or decrement to the sort key; or ROWS, to indicate a physical definition of the aggregation group, by counting rows forward or backward from the current row.

— A starting row, which may be the first row of the partition, the current row, or some row determined by a logical or physical offset from the current row.

— An ending row, which may be the last row of the partition, the current row, or some row determined by a logical or physical offset from the current row.

A window is described by a *window descriptor*, consisting of:

— The window name.

— Optionally, the name of another window, called the *ordering window*, that is used to define the partitioning and ordering of the present window.

— The window's <window partition clause> (if not specified, a zero-length string).

— The window's <window order clause> (if not specified, a zero-length string).

— The window's <window aggregation group clause> (if not specified, a zero-length string).

Two <OLAP function>s are computed using the same (possibly non-deterministic) ordering of the rows if any of the following are true:

— The <OLAP function>s identify the same window descriptor.

— The <OLAP function>s window descriptors have <window partition clause>s that enumerate the same number of column references, and those column references are pairwise equivalent in their order of occurence; and their window descriptors have <window order clause>s with the same number of <sort key>s, and those <sort key>s are all column references, and those column references are pairwise equivalent in their order of occurrence.

— The window descriptor of one <OLAP function> is the ordering window of the other <OLAP function>, or both window descriptors identify the same ordering window.

4.2 SQL-invoked routines

Insert after the sixth paragraph An <OLAP function> *OF* is said to be *dependent* on an SQL-invoked routine *SR* if and only if all the following are true:

— *SR* is the ordering function included in the user-defined descriptor of a user-defined type *UDT*.

— Either there is a column reference whose declared type is *UDT* simply contained in the <window partition clause> of the window descriptor referenced by *OF*, or there is a <value expression> whose declared type is *UDT* simply contained in the <window order clause> of the window descriptor referenced by *OF*.

5 Lexical elements

5.1 <token> and <separator>

Function

Specify lexical units (tokens and separators) that participate in SQL language.

Format

```
<non-reserved word> ::=
```

ANSI Only---caused by ISO changes not yet considered by ANSI

```
  !! All alternatives from ANSI X3.135.2
| !! All alternatives from ANSI X3.135.5
```

ISO Only---caused by ANSI changes not yet considered by ISO

```
  !! All alternatives from ISO/IEC 9075-2
| !! All alternatives from ISO/IEC 9075-5
```

```
      | DENSERANK

      | FOLLOWING

      | NULLS

      | OVER

      | PARTITION | PRECEDING

      | RANGE | RANKnul1 | ROWNUMBER

      | TIES

      | UNBOUNDED

<reserved word> ::=
```

ANSI Only---caused by ISO changes not yet considered by ANSI

```
  !! All alternatives from ANSI X3.135.2
| !! All alternatives from ANSI X3.135.5
```

ISO Only---caused by ANSI changes not yet considered by ISO

```
  !! All alternatives from ISO/IEC 9075-2
| !! All alternatives from ISO/IEC 9075-5
```

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

5.1.1 Names and identifiers

Function

Specify names.

Format

```
<window name> ::= <identifier>
```

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

1) Insert this GR A <window name> identifies a window.

Conformance Rules

1) Insert this CR Without Feature T612, “Advanced OLAP functions”, conforming SQL language shall not specify <window name>.

6 Scalar expressions

6.1 <set function specification>

Function

Specify a value derived by the application of a function to an argument.

Format

```
<set function specification> ::=
        <aggregate function>
      | <grouping operation>
```

Delete the definition of <general set function>

Delete the definition of <set function type>

Delete the definition of <computational operation>

Delete the definition of <set quantifier>

NOTE 5 – The deleted definitions are now placed in Subclause 8.1, "<aggregate function>".

Syntax Rules

1) Delete SR 1) through 3)

2) Insert after SR5)a) A <window clause>.

3) Insert this SR If <aggregate function> is specified, then the declared type of the result is the declared type of the <aggregate function>. If the declared type is character string, then the collating sequence and coercibility characteristic are the collating sequence and coercibility characteristic of the <aggregate function>.

4) Delete SRs 6) through 14)

5) Delete SR16)

Access Rules

No additional Access Rules.

General Rules

1) Delete GR1) and GR2)

2) Delete GRs 3)a) through 3)g)

3) Append this GR If <aggregate function> is specified, then the result is the value of the <aggregate function>.

Conformance Rules

1) Delete CRs 1) through 6)

2) Delete CRs 8) and 9)

6.2 <OLAP function>

Function

Specify an OLAP function.

Format

```
<OLAP function> ::=
       <OLAP function type> OVER <window name or specification>

<OLAP function type> ::=
         RANKnul1 ( )
       | DENSERANK ( )
       | ROWNUMBER ( )
       | <aggregate function>

<window name or specification> ::=
         <window name>
       | <in-line window specification>

<in-line window specification> ::=
       <window specification>
```

Syntax Rules

1) Let *OF* be the <OLAP function>.

2) Case:

   a) If *OF* is contained an <order by clause>, then the <order by clause> shall be contained in a <cursor specification> that is a simple table query. Let *ST* be the sort table that is obtained by applying the syntactic transformation of a simple table query, as specified in Subclause 11.1, "<declare cursor>". Let *TE* be the <table expression> contained in the result of that syntactic transformation.

   b) Otherwise, *OF* shall be contained in a <select list> that is immediately contained in a <query specification> *QS* or a <select statement: single row> *SSSR*. Let *QSS* be the innermost <query specification> contained in *QS* that contains *OF*. Let *TE* be the <table expression> immediately contained in *QSS* or *SSSR*.

3) *OF* shall not contain an outer reference or a <subquery>.

4) Let WNS be the <window name or specification>. Let *WDX* be a window descriptor that describes the window defined by *WNS*.

5) If RANK*nul1*, DENSERANK, or ROWNUMBER is specified, then:

   a) The <window order clause> *WOC* of *WDX* shall not be a zero-length string.

   b) The <window aggregation group clause> of *WDX* shall be a zero-length string.

   c) Case:

      i) If *WNS* is a <window name>, then let *WNS1* be *WNS*.

ii) Otherwise, let *WNS1* be the <window specification details> contained in *WNS*.

d) RANK*nul1* ( ) OVER *WNS* is equivalent to the <OLAP function>:

```
( COUNT (*) OVER (WNS1 RANGE UNBOUNDED PRECEDING)
- COUNT (*) OVER (WNS1 RANGE CURRENT ROW) + 1 )
```

e) If DENSERANK is specified, then:

i) Let VE_1, . . . , VE_N be an enumeration of the <value expression>s that are <sort key>s simply contained in *WOC*.

ii) DENSERANK ( ) OVER *WNS* is equivalent to the <OLAP function>:

```
COUNT (DISTINCT ROW ( VE1, . . . , VEN ) )
OVER (WNS1 RANGE UNBOUNDED PRECEDING)
```

f) ROWNUMBER ( ) OVER *WNS* is equivalent to the <OLAP function>:

```
COUNT (*) OVER (WNS1 ROWS UNBOUNDED PRECEDING)
```

6) If <aggregate function> is specified and the <window order clause> of *WDX* is not a zero-length string, then the <window aggregation group clause> of *WDX* shall not be a zero-length string.

7) Let *SL* be the <select list> that simply contains *OF*.

NOTE 6 – If *OF* is originally contained in an <order by clause> of a cursor that is a simple table query, the syntactic transformation of Subclause 11.1, "<declare cursor>", must be applied prior to this rule.

8) Let *SQ* be the <set quantifier> of the <query specification> or <select statement: single row> that simply contains *SL*. If there is no <set quantifier>, then let *SQ* be a zero-length string.

9) If <in-line window specification> is specified, then:

a) Let *WS* be the <window specification>.

b) Let *WSN* be an implementation-dependent <window name> that is not equivalent to any other <window name> in the <table expression> or <select statement: single row> that simply contains *WS*.

c) Let *OFT* be the <OLAP function type>.

d) Let *SLNEW* be the <select list> that is obtained from *SL* by replacing *OF* by:

```
OFT OVER WSN
```

e) Let *FC*, *WC*, *GBC*, and *HC* be <from clause>, <where clause>, <group by clause>, and <having clause>, respectively, of *TE*. If any of <where clause>, <group by clause>, or <having clause> is missing, then let *WC*, *GBC*, or *HC*, respectively, be a zero-length string.

f) Case:

i) If there is no <window clause> simply contained in *TE*, then let *WICNEW* be:

```
WINDOW WSN AS WS
```

ii) Otherwise, let *WIC* be the <window clause> simply contained in *TE* and let *WICNEW* be:

```
WIC, WSN AS WS
```

g) Let *TENEW* be:

```
FC WC GBC HC WICNEW
```

h) Case:

i) If *OF* is simply contained in a <query specification>, then that <query specification> is equivalent to:

```
SELECT SQ SLNEW TENEW
```

ii) Otherwise, *OF* is simply contained in a <select statement: single row>. Let *STL* be the <select target list> of that <select statement: single row>. The <select statement: single row> is equivalent to:

```
SELECT SQ SLNEW INTO STL TENEW
```

Access Rules

None.

General Rules

a) The value of <OLAP function> is the value of the <aggregate function>.

Conformance Rules

1) Without Feature T611, "Elementary OLAP functions", conforming SQL language shall not specify an <OLAP function>.

2) Without Feature T612, "Advanced OLAP functions", conforming SQL language shall not specify <window name>.

3) Without Feature T612, "Advanced OLAP functions", an <aggregate function> simply contained in an <OLAP function> shall not specify DISTINCT if the <window aggregation group clause> of the window descriptor that describes the <window name or specification> is not a zero-length string.

6.3 <value expression>

Function

Specify a value.

Format

```
<nonparenthesized value expression primary> ::=
         !! All alternatives from ISO/IEC 9075-2
       | <OLAP function>
```

Syntax Rules

1) Replace SR 2) The declared type of a <value expression primary> is the declared type of the simply contained <unsigned value specification>, <column reference>, <set function specification>, <scalar subquery>, <case expression>, <value expression>, <cast specification>, <subtype treatment>, <attribute or method reference>, <reference resolution>, <collection value constructor>, <field reference>, <element reference>, <method invocation>, or <static method invocation>, or <OLAP function>, or the effective returns type of the immediately contained <routine invocation>, respectively.

Access Rules

None.

General Rules

1) Replace GR 5) The value of a <value expression primary> is the value of the simply contained <unsigned value specification>, <column reference>, <set function specification>, <rank function specification>, <scalar subquery>, <case expression>, <value expression>, <cast specification>, <subtype treatment>, <collection value constructor>, <field reference>, <element reference>, <method invocation>, <static method invocation>, <routine invocation>, <attribute or method reference> or <OLAP function>.

Conformance Rules

1) Insert this CR Without Feature T611, "Elementary OLAP functions", a <value expression> shall not be an <OLAP function>.

7 Query expressions

7.1 <table expression>

Format

```
<table expression> ::=
      <from clause>
      [ <where clause> ]
      [ <group by clause> ]
      [ <having clause> ]
      [ <window clause> ]
```

Syntax Rules

1) Replace SR 1) The result of a <table expression> is a derived table, whose row type *RT* is the row type of the result of the application of the last of the immediately contained <from clause>, <where clause>, <group by clause> or <having clause> specified in the <table expression>, together with the window descriptors defined by the <window clause>, if specified.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

1) Insert this CR Without Feature T612, "Advanced OLAP functions", conforming SQL language shall not specify a <window clause>.

7.2 <joined table>

Function

Reference a table.

Format

!! *No additional Format items*

Syntax Rules

1) Insert after SR 5) The <search condition> shall not contain an <OLAP function> without an intervening <subquery>.

Access Rules

No additional Access Rules

General Rules

No additional General Rules

Conformance Rules

No additional Conformance Rules.

7.3 <where clause>

Function

Specify a table derived by the application of a <search condition> to the result of the preceding <from clause>.

Format

!! *No additional Format items*

Syntax Rules

1) Insert after SR 2) The <search condition> shall not contain an <OLAP function> without an intervening <subquery>.

Access Rules

No additional Access Rules

General Rules

No additional General Rules

Conformance Rules

No additional Conformance Rules

7.4 <having clause>

Function

Specify a grouped table derived by the elimination of groups that do not satisfy a <search condition>.

Format

!! *No additional Format items*

Syntax Rules

1) Insert after SR 4) The <search condition> shall not contain an <OLAP function> without an intervening <subquery>

Access Rules

No additional Access Rules

General Rules

No additional General Rules

Conformance Rules

No additional Conformance Rules.

7.5 <window clause>

Function

Specify one or more window definitions.

Format

```
<window clause> ::=
       WINDOW <window definition list>

<window definition list> ::=
       <window definition> [ { <comma> <window definition> }... ]

<window definition> ::=
       <new window name> AS <window specification>

<new window name> ::= <window name>

<window specification> ::=
       <left paren> <window specification details> <right paren>

<window specification details> ::=
       [ <existing window name> ]
       [ <window partition clause> ]
       [ <window order clause> ]
       [ <window aggregation group clause> ]

<existing window name> ::= <window name>

<window partition clause> ::=
       PARTITION BY <window partition column reference list>

<window partition column reference list> ::=
       <window partition column reference>
           [ { <comma> <window partition column referenc> }... ]

<window partition column reference> ::=
       <column reference> [ <collate clause> ]

<window order clause> ::=
       ORDER BY <sort specification list>

<window aggregation group clause> ::=
       <window aggregation group units>
       <window aggregation group extent>

<window aggregation group units> ::=
         ROWS
       | RANGE

<window aggregation group extent> ::=
         <window aggregation group start>
       | <window aggregation group between>

<window aggregation group start> ::=
         UNBOUNDED PRECEDING
       | <window aggregation group preceding>
       | CURRENT ROW
```

7.5 <window clause>

```
<window aggregation group preceding> ::=
       <unsigned value specification> PRECEDING

<window aggregation group between> ::=
       BETWEEN <window aggregation group bound 1>
           AND <window aggregation group bound 2>

<window aggregation group bound 1> ::=
       <window aggregation group bound>

<window aggregation group bound 2> ::=
       <window aggregation group bound>

<window aggregation group bound> :=
         <window aggregation group start>
       | UNBOUNDED FOLLOWING
       | <window aggregation group following>

<window aggregation group following> ::=
       <unsigned value specification> FOLLOWING
```

Syntax Rules

1) Let *TE* be the <table expression> that immediately contains the <window clause>.

2) <new window name> *NWN1* shall not be contained in the scope of another <new window name> *NWN2* such that *NWN1* and *NWN2* are equivalent.

3) Let *WDEF* be a <window definition>.

4) Each <column reference> contained in the <window partition clause> or <window order clause> of *WDEF* shall unambiguously reference a column of the derived table that is the result of *TE*. A column referenced in a <window partition clause> is a partitioning column.

   NOTE 7 – If *TE* is a grouped table, then the <column reference>s contained in <window partition clause> or <window order clause> must reference columns of the grouped table obtained by performing the syntactic transformation in Subclause 7.6, "<query specification>".

5) A <window clause> shall not contain an <OLAP function> without an intervening <subquery>.

6) If *WDEF* specifies <window aggregation group between>, then:

   a) <window aggregation group bound 1> shall not specify UNBOUNDED FOLLOWING.

   b) <window aggregation group bound 2> shall not specify UNBOUNDED PRECEDING.

   c) If <window aggregation group bound 1> specifies CURRENT ROW, then <window aggregation group bound 2> shall not specify <window aggregation group preceding>.

   d) If <window aggregation group bound 1> specifies <window aggregation group following>, then <window aggregation group bound 2> shall not specify <window aggregation group preceding> or CURRENT ROW.

7) If *WDEF* specifies <window aggregation group extent>, and does not specify <window aggregation group between>, then let *WAGS* be the <window aggregation group start>. The <window aggregation group extent> is equivalent to

BETWEEN *WAGS* AND CURRENT ROW

8) If *WDEF* specifies an <existing window name> *EWN*, then:

   a) *WDEF* shall be within the scope of a <window name> that is equivalent to <existing window name>.

   b) Let *WDX* be the window descriptor identified by *EWN*.

   c) *WDEF* shall not specify <window partition clause>.

   d) If *WDX* has a <window order clause> that is not a zero-length string, then *WDEF* shall not specify <window order clause>.

   e) *WDX* shall not have a <window aggregation group clause> that is not a zero-length string.

9) If *WDEF*'s <window aggregation group clause> specifies <window aggregation group preceding> or <window aggregation group following>, then let *UVS* be the <unsigned value specification> simply contained in the <window aggregation group preceding> or <window aggregation group following>.

   Case:

   a) If RANGE is specified, then *WDEF*'s <window order clause> shall contain a single <sort key> *SK*. The declared type of *SK* shall be numeric, datetime, or interval. The declared type of *UVS* shall be numeric if the declared type of *SK* is numeric; otherwise, it shall be an interval type that may be added to or subtracted from the declared type of *SK* according to the Syntax Rules of Subclause 6.28, "<datetime value expression>", and Subclause 6.29, "<interval value expression>", in
   ANSI ANSI X3.135.2.
   ISO ISO/IEC 9075-2.

   b) If ROWS is specified, then the declared type of *UVS* shall be exact numeric with scale 0 (zero).

10) The scope of the <new window name> simply contained in *WDEF* consists of any <window definition>s that follow *WDEF* in the <window clause>, together with the <select list> of the <query specification> or <select statement: single row> that simply contains the <window clause>. If the <window clause> is simply contained in a <query specification> that is the <query expression body> of a <declare cursor> that is a simple table query, then the scope of <new window name> also includes the <order by clause> of the <declare cursor>.

11) Two window descriptors *WD1* and *WD2* are *order-equivalent* if all of the following conditions are met:

   a) Let $WPCR1_i$, 1 (one) $\leq i \leq N1$, and $WPCR2_i$, 1 (one) $\leq i \leq N2$, be enumerations of the <window partition column reference>s contained in the <window partition clause> of *WD1* and *WD2*, respectively, in order from left to right. $N1 = N2$, and, for all i, $WPCR1_i$ and $WPCR2_i$ are equivalent column references.

   b) Let $SS1_i$, 1 (one) $\leq i \leq M1$, and $SS2_i$, 1 *one* $\leq i \leq M2$, be enumerations of the <sort specification>s contained in the <window order clause> of *WD1* and *WD2*, respectively, in order from left to right. $M1 = M2$, and, for all i, $SS1_i$ and $SS2_i$ contain <sort key>s that are equivalent column references, specify or imply the same <ordering specification>, specify or imply the same <collate clause>, if any, and specify or imply the same <null ordering>.

7.5 <window clause>

Access Rules

None.

General Rules

1) Let *TE* be the <table expression> that simply contains the <window clause>. Let *SL* be the <select list> of the <query specification> or <select statement: single row> that immediately contains *TE*.

   Case:

   a) If *SL* has no <OLAP function>, then the <window clause> is disregarded, and the result of *TE* is the result of the last <from clause>, <where clause>, <group by clause> or <having clause> of *TE*.

   b) Otherwise, let *RTE* be the result of the last <from clause> or <where clause> simply contained in *TE*.

      NOTE 8 – Although it is permissible to have a <group by clause> or a <having clause> with a <window clause>, if there are any <OLAP function>s, then the <group by clause> and <having clause> are removed by a syntactic transformation in Subclause 7.6, "<query specification>", and so are not considered here.

      i) A window descriptor *WDESC* is created for each <window definition> *WDEF*, as follows:

         1) *WDESC*'s window name is the <new window name> simply contained in *WDEF*.

         2) Let *EWN* be the <existing window name> simply contained in *WDEF*, if any. Let *WDX* be the window descriptor identified by *EWN* if <existing window name> is specified.

         3) The name of the ordering window of *WDESC* is *EWN*, if <existing window name> is specified and if the <window order clause> of *WDX* is not a zero-length string; otherwise, there is no ordering window name.

         4) *WDESC*'s <window partition clause> is the <window partition clause> simply contained in *WDEF*, if one is specified; otherwise it. is the <window partition clause> of *WDX*, if <existing window name> is specified; otherwise, it is a zero-length string.

         5) *WDESC*'s <window order clause> is the <window order clause> simply contained in *WDEF*, if one is specified; otherwise, it is the <window order clause> of *WDX*, if <existing window name> is specified; otherwise it is a zero-length string.

         6) *WDESC*'s <window aggregation group clause> is the <window aggregation group clause> simply contained in *WDEF*, if <window aggregation group clause> is specified; otherwise it is a zero-length string.

      ii) The result of <window clause> is *RTE*, together with the window descriptors defined by the <window clause>.

2) Let *WD* be a window descriptor.

3) *WD* defines, for each row *R* of *RTE*, the partition of *R* under *WD*, consisting of the multiset of rows of *RTE* that are not distinct from *R* in the partitioning columns of *WD*. If the <window partition clause> is a zero-length string, then the partition of *R* is the entire result *RTE*.

4) *WD* also defines an ordering of the rows of each partition defined by *WD*, according to the General Rules of Subclause 8.2, "<sort specification list>", using the <sort specification list> simply contained in *WD*'s <window order clause>. If the <window order clause> is a zero-length string, then the ordering is entirely implementation-dependent, and all rows are peers. Although the ordering of peer rows within a partition is implementation-dependent, the ordering shall be the same for all window descriptors that are order-equivalent. It shall also be the same for any pair of windows *W1* and *W2* such that *W1* is the ordering window for *W2*.

5) *WD* also defines for each row *R* of *RTE* the aggregation group *AG* of *R*, consisting of a multiset of rows. Let *WAGC* be the <window aggregation group clause> of *WD*. *AG* is defined as follows:

   a) If *WAGC* is a zero-length string, then *AG* is the partition of R.

   b) Otherwise, let *AG* initially be the partition of *R* defined by *WD*. Let *WAGB1* be the <window aggregation group bound 1> and let *WAGB2* be the <window aggregation group bound 2> contained in *WAGC*.

      Case:

      i) If RANGE is specified, then:

         1) Case:

            NOTE 9 – In the following subrules, if *WAGB1* specifies UNBOUNDED PRECEDING, then no rows are removed from *AG* by this step. *WAGB1* may not be UNBOUNDED FOLLOWING.

            A) If *WAGB1* specifies <window aggregation group preceding>, then let *V1P* be the value of the <unsigned value specification>.

               I) If *V1P* is negative or the null value, then an exception condition is raised: *data exception—invalid preceding or following size in OLAP function*.

               II) Otherwise, let *SK* be the only <sort key> contained in the <window order clause> of *WD*. Let *VSK* be the value of *SK* for the current row.

                  Case:

                  1) If *VSK* is the null value and if NULLS LAST is specified or implied, then remove from *AG* all rows *R2* such that the value of *SK* in row *R2* is not the null value.

                  2) If *VSK* is not the null value, then:

                     a) If NULLS FIRST is specified or implied, then remove from *AG* all rows *R2* such that the value of *SK* in row *R2* is the null value.

                     b) Case:

                        i) If the <ordering specification> contained in the <window order clause> specifies DESC, then let *BOUND* be the value *VSK+V1P*. Remove from *AG* all rows *R2* such that the value of *SK* in row *R2* is greater than *BOUND*.

                        ii) Otherwise, let *BOUND* be the value *VSK–V1P*. Remove from *AG* all rows *R2* such that the value of *SK* in row *R2* is less than *BOUND*.

B) If *WAGB1* specifies CURRENT ROW, then remove from *AG* all rows that are not peers of the current row and that precede the current row in the ordering defined by *WD*.

C) If *WAGB1* specifies <window aggregation group following>, then let *V1F* be the value of the <unsigned value specification>.

I) If *V1F* is negative or the null value, then an exception condition is raised: *data exception—invalid preceding or following size in OLAP function*.

II) Otherwise, let *SK* be the only <sort key> contained in the <window order clause> of *WD*. Let *VSK* be the value of *SK* for the current row.

Case:

1) If *VSK* is the null value and if NULLS LAST is specified or implied, then remove from *AG* all rows *R2* such that the value of *SK* in row *R2* is not the null value.

2) If *VSK* is not the null value, then:

a) If NULLS FIRST is specified or implied, then remove from *AG* all rows *R2* such that the value of *SK* in row *R2* is the null value.

b) Case:

i) If the <ordering specification> contained in the <window order clause> specifies DESC, then let *BOUND* be the value *VSK–V1F*. Remove from *AG* all rows *R2* such that the value of *SK* in row *R2* is greater than *BOUND*.

ii) Otherwise, let *BOUND* be the value *VSK+V1F*. Remove from *AG* all rows *R2* such that the value of *SK* in row *R2* is less than *BOUND*.

2) Case:

NOTE 10 – In the following subrules, if *WAGB2* specifies UNBOUNDED FOLLOWING, then no rows are removed from *WAGB2* by this step. *WAGB2* may not be UNBOUNDED PRECEDING.

A) If *WAGB2* specifies <window aggregation group preceding>, then let *V2P* be the value of the <unsigned value specification>.

I) If *V2P* is negative or the null value, then an exception condition is raised: *data exception—invalid preceding or following size in OLAP function*.

II) Otherwise, let *SK* be the only <sort key> contained in the <window order clause> of *WD*. Let *VSK* be the value of *SK* for the current row.

Case:

1) If *VSK* is the null value and if NULLS FIRST is specified or implied, then remove from *AG* all rows *R2* such that the value of *SK* in row *R2* is not the null value.

2) If *VSK* is not the null value, then:

a) If NULLS LAST is specified or implied, then remove from *AG* all rows *R2* such that the value of *SK* in row *R2* is the null value.

b) Case:

i) If the <ordering specification> contained in the <window order clause> specifies DESC, then let *BOUND* be the value *VSK+V2P*. Remove from *AG* all rows *R2* such that the value of *SK* in row *R2* is less than *BOUND*.

ii) Otherwise, let *BOUND* be the value *VSK–V2P*. Remove from *AG* all rows *R2* such that the value of *SK* in row *R2* is greater than *BOUND*.

B) If *WAGB2* specifies CURRENT ROW, then remove from *AG* all rows following the current row in the ordering defined by *WD* that are not peers of the current row.

C) If *WAGB2* specifies <window aggregation group following>, then let *V2F* be the value of the <unsigned value specification>.

I) If *V2F* is negative or the null value, then an exception condition is raised: *data exception—invalid preceding or following size in OLAP function*.

II) Otherwise, let *SK* be the only <sort key> contained in the <window order clause> of *WD*. Let *VSK* be the value of *SK* for the current row.

Case:

1) If *VSK* is the null value and if NULLS FIRST is specified or implied, then remove from *AG* all rows *R2* such that the value of *SK* in row *R2* is not the null value.

2) If *VSK* is not the null value, then:

a) If NULLS LAST is specified or implied, then remove from *AG* all rows *R2* such that the value of *SK* in row *R2* is the null value.

b) Case:

i) If the <ordering specification> contained in the <window order clause> specifies DESC, then let *BOUND* be the value *VSK–V2F*. Remove from *AG* all rows *R2* such that the value of *SK* in row *R2* is less than *BOUND*.

ii) Otherwise, let *BOUND* be the value *VSK+V2F*. Remove from *AG* all rows *R2* such that the value of *SK* in row *R2* is greater than *BOUND*.

ii) Otherwise, *ROWS* is specified.

1) Case:

NOTE 11 – In the following subrules, if *WAGB1* specifies UNBOUNDED PRECEDING, then no rows are removed from *AG* by this step. *WAGB1* may not be UNBOUNDED FOLLOWING.

A) If *WAGB1* specifies <window aggregation group preceding>, then let *V1P* be the value of the <unsigned value specification>.

I) If *V1P* is negative or the null value, then an exception condition is raised: *data exception—invalid preceding or following size in OLAP function*.

II) Otherwise, remove from *AG* all rows that are more than *V1P* rows preceding the current row in the ordering defined by *WD*.

B) If *WAGB1* specifies CURRENT ROW, then remove from *AG* all rows that precede the current row in the ordering defined by *WD*.

NOTE 12 – This step removes any peers of the current row that precede it in the implementation-dependent ordering.

C) If *WAGB1* specifies <window aggregation group following>, then let *V1F* be the value of the <unsigned value specification>.

I) If *V1F* is negative or the null value, then an exception condition is raised: *data exception—invalid preceding or following size in OLAP function*.

II) Otherwise, remove from *AG* all rows that precede the current row and all rows that are less than *V1F* rows following the current row in the ordering defined by *WD*.

NOTE 13 – If *V1F* is zero, then the current row is not removed from *AG* by this step; otherwise, the current row is removed from *AG*.

2) Case:

NOTE 14 – In the following subrules, if WAGB2 specifies UNBOUNDED FOLLOWING, then no rows are removed from *AG* by this step. *WAGB2* may not be UNBOUNDED PRECEDING.

A) If *WAGB2* specifies <window aggregation group preceding>, then let *V2P* be the value of the <unsigned value specification>.

I) If *V2P* is negative or the null value, then an exception condition is raised: *data exception—invalid preceding or following size in OLAP function*.

II) Otherwise, remove from *AG* all rows that follow the current row and all rows that are less than *V2P* rows preceding the current row in the ordering defined by *WD*.

NOTE 15 – If *V2P* is zero, then the current row is not removed from *AG* by this step; otherwise, the current row is removed from *AG*.

B) If *WAGB2* specifies CURRENT ROW, then remove from *AG* all rows that follow the current row in the ordering defined by *WD*.

NOTE 16 – This step removes any peers of the current row that follow it in the implementation-dependent ordering.

C) If *WAGB2* specifies <window aggregation group following>, then let *V2F* be the value of the <unsigned value specification>.

I) If *V2F* is negative or the null value, then an exception condition is raised: *data exception—invalid preceding or following size in OLAP function*.

II) Otherwise, remove from *AG* all rows that are more than *V2F* rows following the current row in the ordering defined by *WD*.

Conformance Rules

1) Without Feature T611, "Elementary OLAP functions", conforming SQL language shall not specify <window specification>.

2) Without Feature T612, "Advanced OLAP functions", conforming SQL language shall not specify a <window clause>.

3) Without Feature T612, "Advanced OLAP functions", conforming SQL language shall not specify <existing window name>.

7.6 <query specification>

Function

Specify a table derived from the result of a <table expression>.

Format

!! *No additional Format items*.

Syntax Rules

1) Insert after SR 10) Each column reference contained in an <OLAP function> shall unambiguously reference a column of *T*.

2) Insert this SR If both of the following two conditions are satisfied, then *QS* is a *grouped, windowed query*:

   a) *T* is a grouped table.

   b) Some <derived column> simply contained in *QS* simply contains an <OLAP function>.

3) Insert this SR A grouped, windowed query *GWQ* is transformed to an equivalent <query specification> as follows:

   a) If *GWQ* simply contains a <group by clause> *GBY*, then *GBY* shall not contain ROLLUP, CUBE, or GROUPING SETS.

   b) If *GWQ* contains an <in-line window specification>, then apply the syntactic transformation specified in Subclause 6.2, "<OLAP function>".

   c) If the <select list> of *GWQ* contains <asterisk> or <qualified asterisk>, then apply the syntactic transformations specified in Subclause 7.11, "<query specification>", in
   ANSI ANSI X3.135.2.
   ISO ISO/iEC 9075-2.

   d) Let *GWQ2* be the result of the preceding transformations, if any.

   e) Let *SL*, *FC*, *WC*, *GBC*, *HC*, and *WIC* be the <select list>, <from clause>, <where clause>, <group by clause>, <having clause>, and <window clause>, respectively, of *GWQ2*. If any of <where clause>, <group by clause>, or <having clause>, are missing, then let *WC*, *GBC*, and *HC*, respectively, be a zero-length string. Let *SQ* be the <set quantifier> immediately contained in the <query specification> of *GWQ2*, if any; otherwise, let *SQ* be a zero-length string.

   NOTE 17 – *GWQ2* can not lack a <window clause>, since the syntactic transformation of Subclause 6.2, "<OLAP function>", will create one if there is not one in *GWQ* already.

   f) Let *N1* be the number of <set function specification>s directly contained in *GWQ2*.

   g) Let SFS_i, 1 (one) $\leq i \leq N1$, be an enumeration of the <set function specification>s directly contained in *GWQ2*.

h) Let $SFSI_i$, 1 (one) $\leq i \leq N1$, be a list of <identifier>s that are distinct from each other and distinct from all <identifier>s contained in *GWQ2*.

i) Let *SFSL* be:

SFS_1 AS $SFSI_1$, SFS_2 AS $SFSI_2$, . . . , SFS_{N1} AS $SFSI_{N1}$

j) Let *HCNEW* be obtained from *HC* by replacing each <set function specification> SFS_i by the corresponding <identifier> $SFSI_i$) .

k) Let *N2* be the number of <column reference>s directly contained in *GWQ2*.

l) Let CR_j, 1 (one) $\leq j \leq N2$, be an enumeration of the <column reference>s directly contained in *GWQ2*.

m) Let CRI_j, 1 (one) $\leq j \leq N2$, be a list of <identifier>s that are distinct from each other, distinct from all identifiers in *GWQ2*, and distinct from all $SFSI_i$.

n) Let *CRL* be:

CR_1 AS CR_2 AS CRI_2, . . . , CR_{N2} AS CRI_{N2}

o) Let *N3* be the number of <derived column>s simply contained in *SL* that do not specify <as clause>.

p) Let $DCOL_k$, 1 (one) $\leq k \leq N3$, be the <derived column>s simply contained in *SL* that do not specify an <as clause>. For each *k*, let $COLN_k$ be the <column name> determined as follows:

i) If $DCOL_k$ is a single column reference, then let $COLN_k$ be the <column name> of the column designated by the column reference.

ii) Otherwise, let $COLN_k$ be an implementation-dependent <column name> that is not equivalent to the <column name> of any column, other than itself, of a table referenced by any <table reference> contained in the SQL-statement.

q) Let *SL2* be obtained from *SL* by replacing each <derived column> $DCOL_k$ by

$DCOL_k$ AS $COLN_k$

r) Let *GWQN* be an arbitrary <identifier>.

s) Let *SLNEW* be the <select list> obtained from *SL2* by replacing each <set function specification> SFS_i by $GWQN.SFSI_i$ and replacing each directly contained <column reference> CR_j by $GWQN.CRI_j$.

t) Let *WICNEW* be the <window clause> obtained from *WIC* by replacing each <column reference> CR_j by $GWQN.CRI_j$.

u) *GWQ* is equivalent to the following <query specification>:

7.6 <query specification>

```
SELECT SLNEW
FROM ( SELECT SQ SFSL, CRL
       FC
       WC
       GBC
       HC ) AS GWQN
WICNEW
```

4) Insert after SR 11)c) The <query specification> contains an <OLAP function> that specifies ROWNUMBER or whose associated <window specification> specifies ROWS.

5) Insert after SR 16)a)viii) An <OLAP function> that does not contain RANK*nul1*, DENSERANK, ROWNUMBER, or COUNT.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

8 Additional common elements

8.1 <aggregate function>

Function

Specify a value computed from a multiset of rows.

Format

```
<aggregate function> ::=
        COUNT <left paren> <asterisk> <right paren>
      | <general set function>

<general set function> ::=
      <set function type> <left paren> [ <set quantifier> ]
          <value expression> <right paren>

<set function type> ::=
      <computational operation>

<computational operation> ::=
        AVG | MAX | MIN | SUM
      | EVERY | ANY | SOME
      | COUNT

<set quantifier> ::=
        DISTINCT
      | ALL
```

Syntax Rules

1) Let *AF* be the <aggregate function>.

2) If <set quantifier> is not specified, then ALL is implicit.

3) The argument of COUNT(*) and the argument source of a <general set function> is

   Case:

   a) If *AF* is immediately contained in a <set function specification>, a table or group of a grouped table as specified in Subclause 7.4, "<having clause>", and Subclause 7.6, "<query specification>".

   b) Otherwise, the multiset of rows in the current row's aggregation group defined by the window descriptor identified by the <window aggregate> that simply contains *AF*, as defined in Subclause 7.5, "<window clause>".

4) Let *T* be the argument or argument source of *AF*.

5) The <value expression> shall not contain an <OLAP function>.

6) Let *DT* be the declared type of the <value expression>.

7) If *AF* specifies a <general set function> whose <set qualifier> is DISTINCT and *DT* is a user-defined type, then the comparison form of *DT* shall be FULL.

8) If *AF* specifies a <set function type> that is MAX or MIN and *DT* is a user-defined type, then the comparison form of *DT* shall be FULL.

9) If *AF* specifies a <set function type> that is MAX or MIN, then DT shall not be a <collection type>, row type, reference type, or large object string type.

10) If EVERY, ANY, or SOME is specified, then *DT* shall be boolean and the declared type of the result is boolean.

11) If COUNT is specified, then the declared type of the result is exact numeric with implementation-defined precision and scale of 0 (zero).

12) If MAX or MIN is specified, then the declared type of the result is *DT*.

13) If SUM or AVG is specified, then:

    a) *DT* shall be a numeric type or an interval type.

    b) If SUM is specified and *DT* is exact numeric with scale *S*, then the declared type of the result is exact numeric with implementation-defined precision and scale *S*.

    c) If AVG is specified and *DT* is exact numeric, then the declared type of the result is exact numeric with implementation-defined precision not less than the precision of *DT* and implementation-defined scale not less than the scale of *DT*.

    d) If *DT* is approximate numeric, then the declared type of the result is approximate numeric with implementation-defined precision not less than the precision of *DT*.

    e) If *DT* is interval, then the declared type of the result is interval with the same precision as *DT*.

14) If the declared type of the result is character string, then the collating sequence and the coercibility characteristic are determined as in Subclause 4.2.3, "Rules determining collating sequence usage".

Access Rules

None.

General Rules

1) Case:

    a) If COUNT( ) is specified, then the result is the cardinality of *T*.

    b) Otherwise, let *TX* be the single-column table that is the result of applying the <value expression> to each row of *T* and eliminating null values. If one or more null values are eliminated, then a completion condition is raised: *warning—null value eliminated in set function*.

2) Case:

   a) If DISTINCT is specified, then let *TXA* be the result of eliminating redundant duplicate values from *TX*, using the comparison rules specified in Subclause 8.2, "<comparison predicate>", in
   ANSI ANSI X3.135.2
   ISO ISO/IEC 9075-2
   to identify the redundant duplicate values.

   b) Otherwise, let *TXA* be *TX*.

3) Case:

   a) If the <general set function> COUNT is specified, then the result is the cardinality of *TXA*.

   b) If *TXA* is empty, then the result is the null value.

   c) If AVG is specified, then the result is the average of the values in *TXA*.

   d) If MAX or MIN is specified, then the result is respectively the maximum or minimum value in *TXA*. These results are determined using the comparison rules specified in Subclause 8.2, "<comparison predicate>", in
   ANSI ANSI X3.135.2.
   ISO ISO/IEC 9075-2.
   If *DT* is a user-defined type and the comparison of two values in *TXA* results in *unknown*, then the maximum or minimum of *TXA* is implementation-dependent.

   e) If SUM is specified, then the result is the sum of the values in *TXA*. If the sum is not within the range of the declared type of the result, then an exception condition is raised: *data exception—numeric value out of range*.

   f) If EVERY is specified, then

      Case:

      i) If the value of some element of TXA is *false*, then the result is *false*.

      ii) Otherwise, the result is *true*.

   g) If ANY or SOME is specified, then

      Case:

      i) If the value of some element of *TXA* is *true*, then the result is *true*.

      ii) Otherwise, the result is *false*.

Conformance Rules

1) Without Feature T031, "BOOLEAN data type", conforming SQL language shall not contain a <set function type> that specifies EVERY, ANY, or SOME.

2) Without Feature F561, "Full value expressions", or Feature F801, "Full set function", if a <general set function> specifies DISTINCT, then the <value expression> shall be a column reference.

3) Without Feature F441, "Extended set function support", if a <general set function> specifies or implies ALL, then COUNT shall not be specified.

8.1 <aggregate function>

4) Without Feature F441, "Extended set function support", if a <general set function> specifies or implies ALL, then the <value expression> shall contain a column reference that references a column of *T*.

5) Without Feature F441, "Extended set function support", if the <value expression> contains a column reference that is an outer reference, then the <value expression> shall be a column reference.

6) Without Feature F441, "Extended set function support", no column reference contained in an <aggregate function> shall reference a column derived from a <value expression> that generally contains an <aggregate function> *SFS2* without an intervening <routine invocation>.

7) Without Feature S024, "Enhanced structured types", in a <general set function>, if MAX or MIN is specified, then the <value expression> shall not be of a structured type.

8) Without Feature S024, "Enhanced structured types", the declared type of a <general set function> shall not be structured type.

8.2 <sort specification list>

Function

Specify a sort order.

Format

```
<sort specification list> ::=
      <sort specification> [ { <comma> <sort specification> }... ]

<sort specification> ::=
      <sort key> [ <collate clause> ]
          [ <ordering specification> ] [ <null ordering> ]

<sort key> ::=
      <value expression>

<ordering specification> ::=
        ASC
      | DESC

<null ordering> ::=
        NULLS FIRST
      | NULLS LAST
```

****Editor's Note****

Chris Farrer and Dave Birdsall noted the following Possible Problem: the <collate clause> in the definition of <sort specification> is ambiguous with <collate clause> in <character factor>, since <sort key> ::= <value expression> ::= <string value expression> ::= <character value expression> ::= <character factor>. This problem was not present in SQL-92; it was caused by permitting <sort key> ::= <value expression> in SQL-99. See Possible Problem NNN. This is also a problem in Foundation.

Syntax Rules

1) Let *DT* be the declared type of the <value expression> immediately contained in the <sort key> contained in a <sort specification>.

   a) *DT* shall not be large object string, reference type, or array type.

   b) If the <sort specification> contains a <collate clause>, then *DT* shall be character string.

   c) If *DT* is a user-defined type, then the comparison form of *DT* shall be FULL.

2) 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>.

Access Rules

None.

General Rules

1) A <sort specification list> defines an ordering of rows, as follows:

   a) Let N be the number of <sort specification>s.

   b) Let K_i, 1 (one) $\leq i \leq N$, be the <sort key> contained in the i-th <sort specification>.

   c) Each <sort specification> specifies the *sort direction* for the corresponding sort key K_i. If DESC is not specified in the i-th <sort specification>, then the sort direction for K_i is ascending and the applicable <comp op> is the <less than operator>. Otherwise, the sort direction for K_i is descending and the applicable <comp op> is the <greater than operator>.

   d) Let P be any row of the multiset of rows to be ordered, and let Q be any other row of the same multiset of rows.

   e) Let PV_i and QV_i be the values of K_i in P and Q, respectively. The relative position of rows P and Q in the result is determined by comparing PV_i and QV_i according to the rules of Subclause 8.2, "<comparison predicate>", in
   ANSI ANSI X3.135.2,
   ISO ISO/IEC 9075-2,
   where the <comp op> is the applicable <comp op> for K_i, with the following special treatment of null values.

      Case:

      i) If PV_i and QV_i are both null, then they are considered equal to each other.

      ii) If PV_i is null and QV_i is not null, then

         Case:

         1) If NULLS FIRST is specified or implied, then PV_i <comp op> QV_i is considered to be *true* .

         2) If NULLS LAST is specified or implied, then PV_i <comp op> QV_i is considered to be *false* .

      iii) If PV_i is not null and QV_i is null, then

         Case:

         1) If NULLS FIRST is specified or implied, then PV_i <comp op> QV_i is considered to be *false* .

         2) If NULLS LAST is specified or implied, then PV_i <comp op> QV_i is considered to be *true* .

   f) PV_i is said to *precede* QV_i if the value of the <comparison predicate> "PV_i <comp op> QV_i" is *true* for the applicable <comp op>.

   g) 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.

   h) The relative position of row P is before row Q if PV_n precedes QV_n for some n, 1 (one) $\leq n \leq N$, and PV_i is not distinct from QV_i for all $i < n$.

i) Two rows that are not distinct with respect to the <sort specification>s are said to be *peers* of each other. The relative ordering of peers is implementation-dependent.

Conformance Rules

1) Without Feature T611, "Elementary OLAP functions", conforming SQL language shall not specify <null ordering>.

9 Schema definition and manipulation

9.1 <drop routine statement>

Function

Destroy an SQL-invoked routine.

Format

!! *No additional Format items*

Syntax Rules

1) Replace Syntax Rule 4) If SR is the ordering function included in the user-defined descriptor of any user-defined type *UDT*, then:

   a) Let *P* be a <predicate> that is dependent on *SR*, let *SFS* be a <set function specification> that is dependent on *SR*, let *OF* be an <OLAP function> that is dependent on *SR*, and let *GBC* be a <group by clause> that is dependent on *SR*.

      NOTE 18 – The notion of a <predicate>, <set function specification>, <OLAP function>, or <group by clause> that is dependent on an SQL-invoked routine is defined in Subclause 4.2, "SQL-invoked routines".

   b) If RESTRICT is specified, then neither *P*, *SFS*, *OF*, nor *GBC* shall be contained in any of the following:

      i) The <SQL routine body> of any routine descriptor.

      ii) The <query expression> of any view descriptor.

      iii) The <search condition> of any constraint descriptor or assertion descriptor.

      iv) The triggered action of any trigger descriptor.

      NOTE 19 – If CASCADE is specified, then such referencing objects will be dropped by the execution of the <revoke statement> specified in the General Rules of this Subclause.

****Editor's Note****

The preceding informative Note is false; there are no rules in <revoke statement> to drop such dependent objects. See Possible Problem nnn. This is also a problem in Foundation.

Access Rules

No additional Access Rules

9.1 <drop routine statement>

General Rules

No additional General Rules

Conformance Rules

No additional Conformance Rules.

9.2 <drop user-defined ordering statement>

Function

Destroy a user-defined ordering method.

Format

```
!! No additional Format items
```

Syntax Rules

1) Insert after SR 4) Let *SK* be any <sort key> whose declared type is *UDT* that is simply contained in a <window order clause> contained in an <OLAP function>.

2) Replace SR 5) If RESTRICT is specified, then neither *P* nor *SK* shall be contained in any of the following:

   a) The <SQL routine body> of any routine descriptor.

   b) The <query expression> of any view descriptor.

   c) The <search condition> of any constraint descriptor or assertion descriptor.

   d) The triggered action of any trigger descriptor.

   NOTE 20 – If CASCADE is specified, then such referencing objects will be dropped as specified in the General Rules of this Subclause.

Access Rules

No additional Access Rules.

General Rules

1) Replace GR 1) Let *R* be any SQL-invoked routine that contains *P* or *SK* in its <SQL routine body>. Let *SN* be the specific name of *R*. The following <drop routine statement> is effectively executed without further Access Rule checking:

```
DROP SPECIFIC ROUTINE SN CASCADE
```

2) Replace GR 2) Let *V* be any view that contains *P* or *SK* in its <query expression>. Let *VN* be the <table name> of *V*. The following <drop view statement> is effectively executed without further Access Rule checking:

```
DROP VIEW VN CASCADE
```

3) Replace GR 3) Let *T* be any table that contains *P* or *SK* in the <search condition> of any constraint *C* whose constraint descriptor included in the table descriptor of *T*. Let *TN* be the <table

name> of *T*. Let *TCN* be the <constraint name> of *C*. The following <alter table statement> is effectively executed without further Access Rule checking:

```
ALTER TABLE TN DROP CONSTRAINT TCN CASCADE
```

4) Replace GR 4) Let *A* be any assertion that contains *P* or *SK* in its <search condition>. Let *AN* be the <constraint name> of *A*. The following <drop assertion statement> is effectively executed without further Access Rule checking:

```
DROP ASSERTION AN CASCADE
```

5) Replace GR 5) Let *D* be any domain that contains *P* or *SK* in the <search condition> of any constraint descriptor or in the <default option> included in the domain descriptor of *D*. Let *DN* be the <domain name> of *D*. The following <drop domain statement> is effectively executed without further Access Rule checking:

```
DROP DOMAIN DN CASCADE
```

6) Replace GR 6) Let *T* be any trigger that contains *P* or *SK* in its triggered action. Let *TN* be the <trigger name> of *T*. The following <drop trigger statement> is effectively executed without further Access Rule checking:

```
DROP TRIGGER TN CASCADE
```

Conformance Rules

No additional Conformance Rules.

10 SQL-client modules

10.1 Calls to an <externally-invoked procedure>

Function

Define the call to an <externally-invoked procedure> by an SQL-agent.

Syntax Rules

1) Insert into SR 2)e)

```
DATA_EXCEPTION_INVALID_PRECEDING_OR_FOLLOWING_SIZE_IN_OLAP_FUNCTION:
        constant SQLSTATE_TYPE :="22013";
```

11 Data manipulation

11.1 <declare cursor>

Function

Define a cursor

Format

Delete the definition of <sort specification list>

Delete the definition of <sort specification>

Delete the definition of <sort key>

Delete the definition of <ordering specification>

NOTE 21 – The deleted definitions are now placed in Subclause 8.2, "<sort specification list>".

Syntax Rules

1) Replace SR 18)f)i)2)A)II) *SL* shall not specify SELECT DISTINCT or directly contain one or more <set function specification>s.

2) Delete SR 19)

3) Delete SR 22)

NOTE 22 – The deleted Syntax Rules are now found in Subclause 8.2, "<sort specification list>".

Access Rules

No additional Access Rules

General Rules

1) Replace GR 2) If an <order by clause> is specified, then the ordering of rows of the result is determined by the <sort specification list>. The result table specified by the <cursor specification> is *TS* with all extended sort key columns (if any) removed.

Conformance Rules

No additional Conformance Rules.

11.2 <select statement: single row>

Function

Retrieve values from a specified row of a table.

Format

!! *No additional Format items*

Syntax Rules

1) Delete SR 5)

2) Insert after SR 6) The <select statement: single row> is possibly nondeterministic if *S* is possibly nondeterministic.

Access Rules

No additional Access Rules

General Rules

No additional General Rules

Conformance Rules

No additional Conformance Rules.

12 Dynamic SQL

12.1 <prepare statement>

Function

Retrieve values from a specified row of a table.

Format

!! *No additional Format items*

Syntax Rules

No additional Syntax Rules

Access Rules

No additional Access Rules

General Rules

1) Insert after SR 6)a)xxvii) If *DP* is contained in a <window aggregation group preceding> or a <window aggregation group following> contained in a <window specification> *WS*, then

   Case:

   a) If *WS* specifies ROWS, then *DT* is NUMERIC (*MP*, 0).

   b) Otherwise, let *SDT* be the data type of the single <sort key> contained in *WS*.

      Case:

      i) If *SDT* is a numeric type, then *DT* is *SDT*.

      ii) If *SDT* is DATE, then *DT* is INTERVAL DAY.

      iii) If *SDT* is TIME(*P*) WITHOUT TIME ZONE or TIME(*P*) WITH TIME ZONE, then *DT* is INTERVAL HOUR TO SECOND(*P*).

      iv) If *SDT* is TIMESTAMP(*P*) WITHOUT TIME ZONE or TIMESTAMP(*P*) WITH TIME ZONE, then DT is INTERVAL DAY TO SECOND(*P*).

Conformance Rules

No additional Conformance Rules.

13 Status codes

13.1 SQLSTATE

Table 2—SQLSTATE class and subclass values

ANSI Only—caused by ISO changes not yet considered by ANSI

Category	Condition	Class	Subcondition	Subclass
	All alternatives from ANSI X3.135.2 *All alternatives from ANSI X3.135.5*			

ISO Only—caused by ANSI changes not yet considered by ISO

	All alternatives from ISO/IEC 9075-2 *All alternatives from ISO/IEC 9075-5*			
X	data exception	22	invalid preceding or following size in OLAP function	013

14 Conformance

14.1 General conformance requirements

Insert this paragraph The list of features that are implied by other features is shown in Table 3, "Implied feature relationships".

Table 3—Implied feature relationships

Feature ID	Feature Description	Implied Feature ID	Implied Feature Description
T612	Advanced OLAP functions	T611	Elementary OLAP functions

Annex A
(informative)

SQL conformance summary

****Editor's Note****
The contents of this Annex will be generated automatically.

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

1) To Be Supplied

2) Specifications for Feature , "":

Annex B
(informative)

Implementation-defined elements

Insert this paragraph This Annex references those features that are identified in the body of this amendment to
ANSI ANSI X3.135
ISO ISO/IEC 9075
as implementation-defined. Insert this paragraph The term *implementation-defined* is used to identify characteristics that may differ between SQL-implementations, but shall be defined for each particular SQL-implementation.

1) Subclause 8.1, "<aggregate function>":

   a) If COUNT is specified, then the declared type of the result is exact numeric with implementation-defined precision and 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 exact numeric with implementation-defined precision and scale *S*.

      ii) If AVG is specified and the declared type of the argument is exact numeric, then the declared type of the result is exact numeric with implementation-defined precision not less than the precision of *DT* and implementation-defined scale not less than the scale of *DT*.

      iii) If the declared type of the argument is approximate numeric, then the declared type of the result is approximate numeric with implementation-defined precision not less than the precision of *DT*.

      iv) If the declared type of the argument is interval, then the declared type of the result is interval with the same precision as *DT*.

2) Subclause 8.2, "<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>.

Annex C
(informative)

Implementation-dependent elements

Insert this paragraph This Annex references those places where this amendment to
ANSI X3.135
ISO ISO/IEC 9075
states explicitly that the actions of a conforming SQL-implementation are implementation-dependent. Insert this paragraph The term *implementation-dependent* is used to identify characteristics that may differ between SQL-implementations, but that are not necessarily specified for any particular SQL-implementation.

1) Subclause 4.1.3.1, "Windows":

   a) The window name of a window defined implicitly by an <in-line window specification> is implementation-dependent.

2) Subclause 7.5, "<window clause>":

   a) If the <window order clause> of a <window specification> is a zero-length string, then the ordering is entirely implementation-dependent.

   b) The ordering of peer rows within a partition is implementation-dependent, but the ordering shall be the same for all window descriptors that are order-equivalent. It shall also be the same for windows *W1* and *W2* if *W1* is the ordering window for *W2*.

3) Subclause 7.6, "<query specification>":

   a) The <column name> of a <derived column> that is not a <column reference> and that has no <as clause> is implementation-dependent, but 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.

4) Subclause 8.1, "<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.

5) Subclause 8.2, "<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.

Annex D
(informative)

SQL feature and package taxonomy

This Annex describes a taxonomy of features of the SQL language.

Table 4, "SQL/OLAP feature taxonomy for features outside Core SQL", contains a taxonomy of the features of the SQL language that are specified in this amendment to
ANSI ANSI X3.135.
ISO ISO/IEC 9075.
In this table, 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
ANSI ANSI X3.135
ISO ISO/IEC 9075
without notice.

The second column, "Feature ID", specifies the formal identification of each feature and each subfeature contained in the table. The Feature ID is stable and can be depended on to remain constant. A Feature ID value comprises either a letter and three digits or a letter, three digits, a hyphen, and one or two additional digits. Feature ID values containing a hyphen and additional digits indicate "subfeatures" that help to define complete features, which are in turn indicated by Feature ID values without a hyphen. Only entire features are used to specify the contents of Core SQL and various packages.

The "Feature Name" column contains a brief description of the feature or subfeature associated with the Feature ID value.

Table 4—SQL/OLAP feature taxonomy for features outside Core SQL

	Feature ID	Feature Name
1	T611	Elementary OLAP functions
2	T612	Advanced OLAP functions

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.