INTERNATIONAL STANDARD

ISO/IEC 9075-15

Second edition
2023-06

Information technology — Database languages SQL —

Part 15: Multidimensional arrays (SQL/MDA)

Technologies de l'information — Langages de base de données SQL —

Partie 15: Tableaux multi-dimensionnels (SQL/MDA)

Reference number
ISO/IEC 9075-15:2023(E)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

Tables

Table

Page

Foreword

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.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

ISO and IEC draw attention to the possibility that the implementation of this document may involve the use of (a) patent(s). ISO and IEC take no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, ISO and IEC have not received notice of (a) patent(s) which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at www.iso.org/patents and https://patents.iec.ch. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

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

This second edition cancels and replaces the first edition (ISO/IEC 9075-15:2019), which has been technically revised. It also incorporates the Technical Corrigendum ISO/IEC 9075-15:2019/Cor.1:2022.

The main changes are as follows:

— improve the presentation and accuracy of the summaries of implementation-defined and implementation-dependent aspects of this document;

— introduction of several digital artifacts;

— alignment with updated ISO house style and other guidelines for creating standards.

This second edition of ISO/IEC 9075-15 is designed to be used in conjunction with the following editions of other parts of the ISO/IEC 9075 series, all published in 2023:

— ISO/IEC 9075-1, sixth edition;

— ISO/IEC 9075-2, sixth edition;

— ISO/IEC 9075-3, sixth edition;

— ISO/IEC 9075-4, seventh edition;

— ISO/IEC 9075-9, fifth edition;

— ISO/IEC 9075-10, fifth edition;

— ISO/IEC 9075-11, fifth edition;

— ISO/IEC 9075-13, fifth edition;

— ISO/IEC 9075-14, sixth edition;

— ISO/IEC 9075-16, first edition.

A list of all parts in the ISO/IEC 9075 series can be found on the ISO and IEC websites.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

Introduction

This document was developed in response to industry demand for the ability to store and manipulate data in the form of multidimensional arrays within databases managed using database language SQL.

The organization of this document is as follows:

1) Clause 1, "Scope", specifies the scope of this document.
2) Clause 2, "Normative references", identifies additional standards that, through reference in this document, constitute provisions of this document.
3) Clause 3, "Terms and definitions", defines the notations and conventions used in this document.
4) Clause 4, "Concepts", presents concepts used in the definition of multidimensional arrays.
5) Clause 5, "Lexical elements", defines a number of lexical elements used in the definition of multidimensional arrays.
6) Clause 6, "Scalar expressions", defines a number of scalar expressions used in the definition of multidimensional arrays.
7) Clause 7, "Query expressions", defines the elements of the language that produce rows and tables of data as used in multidimensional arrays.
8) Clause 8, "Predicates", defines the predicates used in the definition of multidimensional arrays.
9) Clause 9, "Additional common rules", specifies the rules for assignments that retrieve multidimensional array data from or store multidimensional array data into SQL-data, and formation rules for set operations.
10) Clause 10, "Additional common elements", defines additional common elements used in the definition of multidimensional arrays.
11) Clause 11, "Schema definition and manipulation", defines facilities for creating and managing a schema.
12) Clause 12, "SQL-client modules", defines SQL-client modules and externally-invoked procedures in the context of multidimensional arrays.
13) Clause 13, "Data manipulation", defines the data manipulation statements.
14) Clause 14, "Additional data manipulation rules", defines additional rules for data manipulation.
15) Clause 15, "Dynamic SQL", defines the facilities for executing SQL-statements dynamically in the context of multidimensional arrays.
16) Clause 16, "Embedded SQL", defines the host language embeddings in the context of multidimensional arrays.
17) Clause 17, "Call-Level Interface specifications", defines facilities for using SQL through a Call-Level Interface.
18) Clause 18, "Information Schema", defines the Information and Definition Schema objects associated with multidimensional arrays.
19) Clause 19, "Definition Schema", defines base tables on which the viewed tables containing schema information depend.
20) Clause 20, "Status codes", defines SQLSTATE values related to multidimensional arrays.

21) Clause 21, "Conformance", defines the criteria for conformance to this document.

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

23) Annex B, "Implementation-defined elements", is an informative Annex. It lists those features for which the body of this document states that the syntax, the meaning, the returned results, the effect on SQL-data and/or schemas, or other aspect is partly or wholly implementation-defined.

24) Annex C, "Implementation-dependent elements", is an informative Annex. It lists those features for which the body of this document states that the syntax, the meaning, the returned results, the effect on SQL-data and/or schemas, or other aspect is partly or wholly implementation-dependent.

25) Annex D, "SQL optional feature taxonomy", is an informative Annex. It identifies the optional features of the SQL language specified in this document by an identifier and a short descriptive name. This taxonomy is used to specify conformance.

26) Annex E, "Deprecated features", is an informative Annex. It lists features that the responsible Technical Committee intends not to include in a future edition of this document.

27) Annex F, "Incompatibilities with ISO/IEC 9075:2016", is an informative Annex. It lists incompatibilities with the previous version of this document.

28) Annex G, "Defect Reports not addressed in this edition of this document", is an informative Annex. It describes the Defect Reports that were known at the time of publication of this document. Each of these problems is a problem carried forward from the previous edition of the ISO/IEC 9075 series. No new problems have been created in the drafting of this edition of this document.

In the text of this document, in Clause 5, "Lexical elements", through Clause 21, "Conformance", Subclauses begin new pages. Any resulting blank space is not significant.

Information technology — Database language SQL —

Part 15:
Multidimensional arrays (SQL/MDA)

1 Scope

This document defines ways in which Database Language SQL can be used in conjunction with multidimensional arrays.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

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

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

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

ISO/IEC 9075-11, *Information technology — Database languages — SQL — Part 11: Information and Definition Schemas (SQL/Schemata)*

Internet Engineering Task Force (IETF) RFC 2046 *Multipurpose Internet Mail Extensions (MIME), Part Two: Media Types*. Edited by: Freed, N. November 1996
Available at: https://tools.ietf.org/html/rfc2046

Internet Engineering Task Force (IETF) RFC 8259 *The JavaScript Object Notation (JSON) Data Interchange Format*. Edited by: Miller, Matthew December 2018
Available at: https://datatracker.ietf.org/doc/rfc8259/

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 9075-1, ISO/IEC 9075-2, and the following apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at `https://www.iso.org/obp`

— IEC Electropedia: available at `https://www.electropedia.org/`

3.1
coordinate
non-empty ordered list of integers

3.2
maximum MD-extent
<of an MD-array type or a site of MD-array type> *MD-extent* (3.6) of an MD-array type

Note 1 to entry: The term "maximum MD-extent" is used for the *MD-extent* (3.6) of an MD-array type because it defines the maximum MD-extent that a value of that MD-array type can have. The *MD-extent* (3.6) of an MD-array value must be within the maximum MD-extent of the value's type.

3.3
MD-array
ordered collection of elements of the same type associated with an *MD-extent* (3.6) where each element is 1:1 associated with some *coordinate* (3.1) within its *MD-extent* (3.6)

Note 1 to entry: A *coordinate* (3.1) is within an *MD-extent* (3.6) if every coordinate value from the integer list is greater than or equal to the lower limit, and less than or equal to the upper limit of the *MD-interval* (3.7) of the *MD-axis* (3.4) at the position in the *MD-extent* (3.6) as the coordinate value has within the *coordinate* (3.1).

3.4
MD-axis
named *MD-interval* (3.7)

3.5
MD-dimension
number of *MD-axes* (3.4) in the *MD-extent* (3.6) of an *MD-array* (3.3)

3.6
MD-extent
non-empty ordered collection of *MD-axes* (3.4) with no duplicate names

3.7
MD-interval
integer interval given by a pair of lower and upper integer limits such that the lower limit is less than or equal to the upper limit; the interval is closed, i.e., both limits are contained in it

4 Concepts

This Clause modifies Clause 4, "Concepts", in ISO/IEC 9075-2.

4.1 Notations and conventions

This Subclause modifies Subclause 4.1, "Notations and conventions", in ISO/IEC 9075-2.

4.1.1 Notations

This Subclause modifies Subclause 4.1.1, "Notations", in ISO/IEC 9075-2.

The notations used in this document are defined in ISO/IEC 9075-1.

The syntax defined in this document is available from the ISO website as a "digital artifact". See `https://standards.iso.org/iso-iec/9075/-15/ed-2/en/` to download digital artifacts for this document. To download the syntax defined in a plain-text format, select the file named `ISO_IEC_9075-15(E)_MDA.bnf.txt`. To download the syntax defined in an XML format, select the file named `ISO_IEC_9075-15(E)_MDA.bnf.xml`.

4.2 Data types

This Subclause modifies Subclause 4.2, "Data types", in ISO/IEC 9075-2.

4.2.1 General introduction to data types

This Subclause modifies Subclause 4.2.1, "General introduction to data types", in ISO/IEC 9075-2.

Insert into the 4th paragraph, in the 1st list item, after the last list item:

— MD-array type: MDARRAY

Insert into the 5th paragraph, after the last list item:

— MD-array types.

4.2.2 Data type terminology

This Subclause modifies Subclause 4.2.4, "Data type terminology", in ISO/IEC 9075-2.

Insert after the 11th paragraph: A data type *TY* is a *collection-containing type* if exactly one of the following is true:

— *TY* is a collection type;

— *TY* is a row type, and the declared type of some field of *TY* is a collection-containing type;

— *TY* is distinct type, and the source type of *TY* is a collection-containing type;

— *TY* is a structured type and the declared type of some attribute of *TY* is a collection-containing type.

Insert into the 11th paragraph, after the last list item:

— A type *T* is *MD-array-ordered* if *T* is *S*-ordered, where *S* is the set of MD-array types.

4.3 Numbers

This Subclause modifies Subclause 4.5, "Numbers", in ISO/IEC 9075-2.

4.3.1 Operations involving numbers

This Subclause modifies Subclause 4.5.3, "Operations involving numbers", in ISO/IEC 9075-2.

Insert into the 1st paragraph, after the last list item:

— <md-array axis index function> (see Subclause 4.5.4.3, "Operators that operate on MD-array values and return numbers") operates on an MD-array argument and an MD-axis name, and returns an integer denoting its index position in the MD-extent of the MD-array.

— <md-array dimension> (see Subclause 4.5.4.3, "Operators that operate on MD-array values and return numbers") operates on an MD-array argument and returns an integer denoting its MD-dimension.

— <md-array lower axis limit> (see Subclause 4.5.4.3, "Operators that operate on MD-array values and return numbers") operates on an MD-array argument and an MD-axis name or index, and returns the lower limit of an MD-axis.

— <md-array upper axis limit> (see Subclause 4.5.4.3, "Operators that operate on MD-array values and return numbers") operates on an MD-array argument and an MD-axis name or index, and returns the upper limit of an MD-axis.

4.4 User-defined types

This Subclause modifies Subclause 4.9, "User-defined types", in ISO/IEC 9075-2.

4.4.1 Distinct types

This Subclause modifies Subclause 4.9.2, "Distinct types", in ISO/IEC 9075-2.

Insert after the 4th paragraph: *CT* shall not be an MD-array type.

4.5 Collection types

This Subclause modifies Subclause 4.12, "Collection types", in ISO/IEC 9075-2.

4.5.1 Introduction to collection types

This Subclause modifies Subclause 4.12.1, "Introduction to collection types", in ISO/IEC 9075-2.

Insert into the 1st paragraph, after the 1st list item:

— MD-arrays

Augment the 2nd paragraph by adding "MDARRAY" the list of alternatives for KC.

Insert after the 2nd paragraph: A maximum MD-extent is mandatory for MD-arrays.

Insert into the 5th paragraph, after the 1st list item:

— MDARRAY (MD-array type).

Insert into the 5th paragraph, after the 2nd list item:

— If *CT* is an MD-array type, the MD-axis descriptors of each dimension of the MD-array.

Insert after the 5th paragraph: An MD-axis descriptor describes a dimension of an MD-array type. The MD-axis descriptor includes:

— The name of the MD-axis.

— The ordinal position of the MD-axis in the MD-extent of the MD-array type.

— The lower limit of the MD-axis.

— The upper limit of the MD-axis.

Insert into the 7th paragraph, after the last list item:

— MD-array-returning external function

4.5.2 MD-arrays

An MD-array *A* is a collection where each element is uniquely identified by a *d*-dimensional coordinate that is an element of the Cartesian product of the *d* MD-intervals of *A*'s MD-extent. *d* is the MD-dimension of *A*.

The MD-extent of *A* is a non-empty array of MD-axis elements. The MD-axis at position *i*, with 1 (one) ≤ *i* ≤ *d*, consists of a name N_i unique within this MD-extent and a closed MD-interval given by a *lower limit* LO_i and an *upper limit* HI_i. A LO_i or HI_i that is the null value in a maximum MD-extent indicates that the MD-axis has no lower or upper limit, respectively.

Each MD-axis of an MD-extent can be uniquely identified through either its name or its position *i* within the MD-extent, where 1 (one) ≤ *i* ≤ *d*. The *extent* of an MD-axis AE_i is defined as the length of the MD-interval of AE_i : $HI_i - LO_i + 1$ (one). An MD-extent *D* is denoted as [N_1 (LO_1 : HI_1), ..., N_d (LO_d : HI_d)]. The *cardinality* of an MD-extent *DC* is the product of the extents of the MD-axes of *D*: $AE_1 \times ... \times AE_d$.

An MD-extent *D*, denoted as [DN_1 (DLO_1 : DHI_1), ..., DN_d (DLO_d : DHI_d)], is *within* another MD-extent *E*, denoted as [EN_1 (ELO_1 : EHI_1), ..., EN_e (ELO_e : EHI_e)], if *d* ≤ *e*, and, for all *i*, 1 (one) ≤ *i* ≤ *d*, all of the following are true:

— ELO_i is the null value or $DLO_i \geq ELO_i$.

— EHI_i is the null value or $DHI_i \leq EHI_i$.

If *D* is within *E* and *d* = *e*, then *D* is *strictly within E*.

If *d* = *e*, DLO_i is not distinct from ELO_i and DHI_i is not distinct from EHI_i, then *D* is *interval-equal* to *E*. If *D* and *E* are interval-equal, and additionally $DN_i = EN_i$, then *D* is *equal* to *E*.

If *d* = *e* and $DN_i = EN_i$, then the *union* of *D* and *E* is defined as [DN_1 (MIN (DLO_1, ELO_1) : MAX (DHI_1, EHI_1)), ..., DN_d (MIN (DLO_d, ELO_d) : MAX (DHI_d, EHI_d))]. The union *U* of a non-empty set of *N* MD-extents D_1, ..., D_N is defined as defined in this list.

— If *N* is 1 (one), then *U* is D_1.

— If *N* is 2, then *U* is the union of D_1 and D_2.

— Otherwise, let *TMPD* be the union of D_1 and D_2. For all *i*, 3 ≤ *i* ≤ *N*, let *TMPD* be the union of *TMPD* and D_i. *U* is *TMPD*.

A site *AS* of MD-array type has a maximum MD-extent *D*. The MD-extent *E* of an MD-array occupying *AS* is constrained to be strictly within *D*.

An *MD-array type* is a <collection type> whose element type *EDT* is not a collection-containing type. If *AT* is some MD-array type with element type *EDT* and maximum MD-extent *D*, then every value of *AT* is an MD-array with element type *EDT* and maximum MD-extent *D*.

Two MD-arrays *A* and *B* are identical if and only if both *A* and *B* have the same MD-extent *D*, and if, for every coordinate *P* in *D*, the element at coordinate *P* in *A* is identical to the element at coordinate *P* in *B*.

4.5.3 Collection comparison and assignment

This Subclause modifies Subclause 4.12.4, "Collection comparison and assignment", in ISO/IEC 9075-2.

Insert after the 4th paragraph: A value of an MD-array type *AT* with MD-extent *D* is assignable to a site of an MD-array type *AS* with maximum MD-extent *E* if *D* is strictly within *E*.

MD-array values are not comparable.

4.5.4 Operations involving MD-arrays

4.5.4.1 Operators that operate on MD-array values and return MD-array values

<md-array subset> is an operation that returns an MD-array value consisting of only those elements from its input MD-array *A* whose coordinates are within the MD-extent defined by the subset. The subset MD-extent can be defined as:

— a list of <md-axis limits named> and <md-axis slice named>, corresponding to specific MD-axes;

— a list of <md-axis limits positional> and <md-axis slice positional>, listed for every MD-axis in turn;

— an <md-extent> function, which allows subsetting an MD-array to the MD-extent of another MD-array.

<md-axis limits named> and <md-axis limits positional> specify lower and upper trim limits for an MD-axis. A wildcard operator * provided in the lower and/or upper trim limits expands to the current limits of the MD-axis. Trimming does not change the MD-dimension of the result MD-array.

<md-axis slice named> and <md-axis slice positional> specify a single coordinate on the selected MD-axis. Slicing reduces the dimension of the result MD-array by one, i.e., for *N* <md-axis slice named> in an <md-array subset> on *A*, the MD-dimension *d* of the result is *d* = MDDIMENSION(*A*) − *N*.

<md-axis limits named> and <md-axis slice named> allow positionally-independent addressing of MD-axes through their unique names, in contrast to <md-axis limits positional> and <md-axis slice positional>.

<md-array value expression> covers the usual arithmetic, comparison, and logical operations where at least one operand is an MD-array. The MD-extents of MD-arrays *A* and *B* in an <md-array value expression> of the form *A op B* shall be equal. The result is an MD-array value with an MD-extent equal to the MD-extent(s) of the input MD-array(s).

<md-array reshape function> is an operation that, given an MD-array with MD-extent *D* and an MD-extent *E* such that MDDIMENSION(*D*) = MDDIMENSION(*E*), returns an MD-array with the same elements as the input MD-array at coordinates within *E* that are also within *D*, plus additional elements with null values at those coordinates within *E* that are not within *D*. The result MD-array value has an MD-extent *E*.

<md-array shift function> is an operation that, given an MD-array with MD-extent *D* and an offset coordinate of MD-dimension equal to MDDIMENSION(*D*), returns an MD-array with an MD-extent *D* and the same elements as the input MD-array, such that each element at coordinate *P* is shifted by the offset coordinate.

<md-array scale function> is an operation that returns an MD-array with the target MD-extent indicated and elements obtained by using nearest-neighbor interpolation to fit the input MD-array elements at coordinates within the MD-extent of the result MD-array.

<md-array absolute value expression> is a function that returns an MD-array with an MD-extent equal to the MD-extent of its input MD-array, where the element at each coordinate in the result is derived by applying <absolute value expression> to the element at the same coordinate in the input.

<md-array natural logarithm> is a function that returns an MD-array with an MD-extent equal to the MD-extent of its input MD-array, where the element at each coordinate in the result is derived by applying <natural logarithm> to the element at the same coordinate in the input.

<md-array common logarithm> is a function that returns an MD-array with an MD-extent equal to the MD-extent of its input MD-array, where the element at each coordinate in the result is derived by applying <common logarithm> to the element at the same coordinate in the input.

<md-array exponential function> is a function that returns an MD-array with an MD-extent equal to the MD-extent of its input MD-array, where the element at each coordinate in the result is derived by applying <exponential function> to the element at the same coordinate in the input.

<md-array square root> is a function that returns an MD-array with an MD-extent equal to the MD-extent of its input MD-array, where the element at each coordinate in the result is derived by applying <square root> to the element at the same coordinate in the input.

<md-array floor function> is a function that returns an MD-array with an MD-extent equal to the MD-extent of its input MD-array, where the element at each coordinate in the result is derived by applying <floor function> to the element at the same coordinate in the input.

<md-array ceiling function> is a function that returns an MD-array with an MD-extent equal to the MD-extent of its input MD-array, where the element at each coordinate in the result is derived by applying <ceiling function> to the element at the same coordinate in the input.

<md-array trigonometric function> is a function that returns an MD-array with an MD-extent equal to the MD-extent of its input MD-array, where the element at each coordinate in the result is derived by applying <trigonometric function> to the element at the same coordinate in the input.

<md-array power function> is a function that returns an MD-array with an MD-extent equal to the MD-extent of its input MD-arrays, where the element at each coordinate in the result is derived by applying <power function> to the element at the same coordinate in the input and the second argument of the function.

<md-array modulus expression> is a function that returns an MD-array with an MD-extent equal to the MD-extent of its input MD-arrays, where the element at each coordinate in the result is derived by applying <modulus expression> to the element at the same coordinate in the input and the second argument of the function.

<md-array concatenation> is an operation that returns the MD-array value made by joining its MD-array value operands in the order given along a specified MD-axis.

4.5.4.2 Operators that operate on MD-array values and return tables

<md-array extent> is an operation that returns the MD-extent *D* of a given MD-array value *A* as a table of cardinality *CAR* equal to the MD-dimension of *A* and four columns. The row with value *i*, 1 (one) ≤ *i* ≤ *CAR*, in the first column contains the name, lower limit, and upper limit of the *i*-th MD-axis in *D* in the second, third, and fourth column, respectively.

<md-array max extent> is an operation that returns the maximum MD-extent *MD* with the declared type of a given MD-array value *A* as a table of cardinality *CAR* equal to the MD-dimension of *A* and four columns. The row with value *i*, 1 (one) ≤ *i* ≤ *CAR*, in the first column contains the name, lower limit, and upper limit of the *i*-th MD-axis in *MD* in the second, third, and fourth column, respectively.

<collection derived table> allows converting an <md-array value expression> *AVE* with element type *ET* and MD-dimension *d* to a table. The resulting table has *d* columns holding the coordinate values of each element, and, if *ETM* is not a row type, one column, or, if *ET* is a row type of degree *N*, *N* columns for the

element values of *AVE*. If WITH ORDINALITY is specified, then the resulting table has an additional ordinality column holding the values from 1 (one) to the cardinality of *AVE*.

4.5.4.3 Operators that operate on MD-array values and return numbers

<md-array axis index function> is an operation that, for a given MD-array and an MD-axis name, returns the index position of the MD-axis with the specified name in the MD-extent of the MD-array.

<md-array dimension> is an operation that returns the dimension of the MD-array as an exact numeric with scale 0 (zero).

<md-array lower axis limit> is an operation that, for a given MD-array and an MD-axis identified by its position or its name, returns the lower limit the MD-axis as an exact numeric with scale 0 (zero).

<md-array upper axis limit> is an operation that, for a given MD-array and an MD-axis identified by its position or its name, returns the upper limit the MD-axis as an exact numeric with scale 0 (zero).

4.5.4.4 Operators that operate on MD-array values and return character strings

<md-array axis name function> is an operation that, for a given MD-array and an MD-axis identified by its position, returns the name of the MD-axis as a character string.

4.5.4.5 Operators that operate on MD-array values and return numbers or Boolean values

<md-array aggregation expression> is an operation that returns an aggregated value obtained from iterating over all coordinates in the MD-extent indicated in its <md-extent alternative>, evaluating the <value expression primary> at each particular coordinate that satisfies the optional <search condition>, and aggregating the result by applying the <md-array aggregation operator>.

<md-array aggregation function> is an operation that returns a value aggregating all elements of the input MD-array. Table 1, "Table aggregation operators", lists all predefined aggregation functions. Let *A* be an MD-array of numeric element type, let *B* be an MD-array of Boolean element type, and let *C* be an MD-array of any element type.

Table 1 — Table aggregation operators

Operation	Meaning
MDSUM(*A*)	Sum of all elements in *A*
MDAVG(*A*)	Average of all elements in *A*
MDMIN(*A*)	Minimum of all elements in *A*
MDMAX(*A*)	Maximum of all elements in *A*
MDCOUNT(*C*)	Number of non-NULL elements in *C*
MDCOUNT_TRUE(*B*)	Number of *True* non-NULL elements in *B*
MDCOUNT_FALSE(*B*)	Number of *False* non-NULL elements in *B*
MDCOUNT_UNKNOWN(*B*)	Number of *Unknown* elements in *B*
MDANY(*B*)	Is there any element in *B* with value true?
MDALL(*B*)	Do all elements in *B* have value true?

4.5.4.6 Operators that operate on MD-array values and return character or binary strings

<md-array encode function> is an operation that accepts an MD-array and returns a character string or a binary string containing the MD-array encoded in the data format indicated by its media type (specified in RFC 2046). MD-arrays can be encoded using JSON (media type "application/json" according to RFC 8259). Support for encoding to further media types is implementation-defined (IA096).

4.5.4.7 Operators that construct new MD-array values

<md-array value constructor by decoding> is an operation that accepts a character string or a binary string and returns an MD-array containing the elements at their respective coordinates, determined by decoding the character or binary string. Such a character or binary string contains a JSON encoding (media type "application/json" RFC 8259) of the MD-array to be decoded. Optional support for decoding from further media types is implementation-defined (IA095).

<md-array value constructor by enumeration> is an operation that returns an MD-array with the MD-extent specified by <md-extent alternative> and elements enumerated by <md-array element list>.

<md-array value constructor by query> returns an MD-array with the MD-extent specified by <md-extent alternative> and elements provided by the result of <table subquery>.

<md-array value constructor by iteration> returns an MD-array with the MD-extent specified by <md-extent alternative> and each element as the result of <md-array element>. The <md-array element> expression may contain MD-axis variables that allow referencing the element's coordinate.

<md-array value constructor by join> performs a join on two or more MD-arrays of equal MD-extents based on their coordinates. An element in the resulting MD-array is a row value constructed from the corresponding elements of each input MD-array, in the order in which they have been specified. The field names of each element in the result can be explicitly specified or implicitly generated.

4.5.4.8 Operators that operate on MD-array values and return MD-array elements

<md-array element reference> is an operation that returns the MD-array element at a specific coordinate within the MD-extent of the MD-array.

4.5.5 MD-axis variables

An <md-array aggregation expression> or an <md-array value constructor by iteration> are operations that specify *MD-axis variables* for every MD-axis in the MD-extent specified by the <md-extent alternative>. The name of each MD-axis variable is equivalent to the name of the corresponding MD-axis, and its value, when referenced by its name, ranges from the lower to the upper limit of the MD-axis.

An MD-axis variable is specified by <md-axis variable>.

Every MD-axis variable has a *scope*. In <md-array value constructor by iteration>, the scope of an MD-axis variable is the immediately contained <md-array element>. In <md-array aggregation expression>, the scope of an MD-axis variable is the immediately contained <value expression primary> and <search condition>, if specified. Two MD-axis variables that are equivalent are nevertheless distinct if they have different scope.

5 Lexical elements

This Clause modifies Clause 5, "Lexical elements", in ISO/IEC 9075-2.

5.1 <token> and <separator>

This Subclause modifies Subclause 5.2, "<token> and <separator>", in ISO/IEC 9075-2.

Function

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

Format

```
<reserved word> ::=
    !! All alternatives from ISO/IEC 9075-2

  | MDAGGREGATE | MDALL | MDANY | MDARRAY | MDAVG | MDAXIS_HIGH | MDAXIS_INDEX | MDAXIS_LOW
  | MDAXIS_NAME | MDAXIS_NAMES | MDCONCAT | MDCOUNT | MDCOUNT_FALSE
  | MDCOUNT_TRUE | MDCOUNT_UNKNOWN | MDDECODE | MDDIMENSION | MDENCODE | MDEXTENT
  | MDJOIN | MDMAX | MDMAX_EXTENT | MDMIN | MDRESHAPE | MDSCALE | MDSHIFT | MDSUM

<non-reserved word> ::=
    !! All alternatives from ISO/IEC 9075-2
  | ELEMENTS
```

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

5.2 Names and identifiers

This Subclause modifies Subclause 5.4, "Names and identifiers", in ISO/IEC 9075-2.

Function

Specify names.

Format

```
<md-axis variable> ::=
  <identifier>
```

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

1) Insert after the last GR: An <md-axis variable> identifies an MD-axis variable associated with an MD-axis whose name is equivalent to the <md-axis variable>.

Conformance Rules

No additional Conformance Rules.

6 Scalar expressions

This Clause modifies Clause 6, "Scalar expressions", in ISO/IEC 9075-2.

6.1 <data type>

This Subclause modifies Subclause 6.1, "<data type>", in ISO/IEC 9075-2.

Function

Specify a data type.

Format

```
<collection type> ::=
    !! All alternatives from ISO/IEC 9075-2
  | <md-array type>

<md-array type> ::=
  <data type> MDARRAY <maximum md-extent>

<maximum md-extent> ::=
    <left bracket or trigraph> <maximum md-axis list> <right bracket or trigraph>
  | <left bracket or trigraph> <maximum md-axis list anonymous> <right bracket or trigraph>

<maximum md-axis list> ::=
  <maximum md-axis> [ { <comma> <maximum md-axis> }... ]

<maximum md-axis list anonymous> ::=
  <maximum md-axis anonymous> [ { <comma> <maximum md-axis anonymous> }... ]

<maximum md-axis> ::=
  <md-axis name> [ <left paren>
      <maximum md-axis lower limit> <colon> <maximum md-axis upper limit>
      <right paren> ]

<maximum md-axis anonymous> ::=
  <maximum md-axis lower limit> <colon> <maximum md-axis upper limit>

<maximum md-axis lower limit> ::=
  <md-axis limit>

<maximum md-axis upper limit> ::=
  <md-axis limit>

<md-axis limit> ::=
    <md-axis limit fixed>
  | <asterisk>

<md-axis limit fixed> ::=
  <signed numeric literal>

<md-axis name> ::=
  <identifier>
```

Syntax Rules

1) Insert after SR 47): The element type of an <array type> shall not be an MD-array type.

2) Insert after SR 47): An <md-array type> *AT* specifies an *MD-array type*. The <data type> *DT* immediately contained in *AT* is the *element type* of the MD-array type. *DT* shall not be a collection-containing type.

   a) If <maximum md-extent> *EU* immediately contains <maximum md-axis list> *MAL* and any <maximum md-axis> immediately contained in *MAL* specifies an <md-axis name> but does not specify <maximum md-axis lower limit> and <maximum md-axis upper limit>, then *EU* is equivalent to *MN*(* : *).

   b) If any <maximum md-axis lower limit> *ALL* simply contained in *AT* is an <asterisk>, then *ALL* is is the null value.

   c) If any <maximum md-axis upper limit> *AUL* simply contained in *AT* is an <asterisk>, then *AUL* is the null value.

   d) The declared type of <md-axis limit fixed> and <md-axis limit> is an implementation-defined (IV201) exact numeric type with scale 0 (zero).

   e) The minimum value of <md-axis limit fixed> is implementation-defined (IL013). <md-axis limit fixed> shall not be smaller than this value.

   f) The maximum value of <md-axis limit fixed> is implementation-defined (IL014). <md-axis limit fixed> shall not be greater than this value.

   g) Case:

      i) If *EU* immediately contains <maximum md-axis list>, then let *d* be the number of <maximum md-axis> simply contained in *EU*.

      ii) Otherwise, let *d* be the number of <maximum md-axis anonymous> simply contained in *EU*.

   h) *d* is the MD-dimension of *AT*. *d* shall be less than or equal to an implementation-defined (IL017) maximum value.

   i) For *i*, 1 (one) ≤ *i* ≤ *d*,

      i) Case:

         1) If *EU* immediately contains <maximum md-axis list>, then let AU_i be the *i*-th <maximum md-axis> simply contained in *EU*. Let MN_i be the value of <md-axis name> immediately contained in AU_i.

         2) Otherwise, let AU_i be the *i*-th <maximum md-axis anonymous> simply contained in *EU* and let SEQ_i be the <unsigned integer> numeric literal with no leading zeros corresponding to *i*. The Syntax Rules of Subclause 9.15, "Indexed name", are applied with *d* as *LASTINDEX*, SEQ_i as *INDEX*, and 'D' as *PREFIX*; let MN_i be the *INDEXEDNAME* returned from the application of those Syntax Rules.

      ii) Let MLO_i be the <maximum md-axis lower limit> and let MHI_i be the <maximum md-axis upper limit> specified or implied by AU_i.

   j) Let *MD* be [MN_1 (MLO_1 : MHI_1), ..., MN_d (MLO_d : MHI_d)]. *MD* is the maximum MD-extent of a site with data type *AT*.

i) For all i, 1 (one) $\leq i \leq d$, if neither MLO_i or MHI_i are the null value, then MLO_i shall be less than or equal to MHI_i.

ii) For all i and j, 1 (one) $\leq i, j \leq d$, $i \neq j$, MN_i shall not be equivalent to MN_j.

iii) The value of MLO_i is the value of the corresponding <maximum md-axis lower limit>.

iv) The value of MHI_i is the value of the corresponding <maximum md-axis upper limit>.

3) Insert after SR 47): The element type of a <multiset type> shall not be an MD-array type.

Access Rules

No additional Access Rules.

General Rules

1) Insert after GR 20)a): If *KC* is MDARRAY, then the collection type descriptor additionally includes a set of MD-axis descriptors, each of which, for 1 (one) $\leq i \leq d$, contains the name MN_i, the ordinal position i, the lower limit MLO_i, and the upper limit MHI_i for the i-th MD-axis of the MD-array type.

Conformance Rules

1) Insert after the last CR: Without Feature A010, "Multi-dimensional array support", conforming SQL language shall not contain an <md-array type>.

6.2 <value expression primary>

This Subclause modifies Subclause 6.3, "<value expression primary>", in ISO/IEC 9075-2.

Function

Specify a value that is syntactically self-delimited.

Format

```
<non-parenthesized value expression primary> ::=
    !! All alternatives from ISO/IEC 9075-2
  | <md-array aggregation expression>
  | <md-axis variable>
  | <md-array element reference>

<collection value constructor> ::=
    !! All alternatives from ISO/IEC 9075-2
  | <md-array value constructor>
```

Syntax Rules

1) Augment SR 1) by adding "<md-array element reference>, <md-axis variable> and <md-array aggregation expression>" to the list of simply contained BNF non-terminals.

2) Insert after the last SR: The declared type of an <md-axis variable> is exact numeric with scale 0 (zero).

3) Augment SR 3) by adding "<md-array value constructor>" to the list of simply contained BNF non-terminals.

Access Rules

No additional Access Rules.

General Rules

1) Augment GR 1) by adding "<md-array element reference>, <md-axis variable> and <md-array aggregation expression>" to the list of simply contained BNF non-terminals.

2) Augment GR 2) by adding "<md-array value constructor>" to the list of simply contained BNF non-terminals.

Conformance Rules

1) Insert after the last CR: Without Feature A010, "Multi-dimensional array support", conforming SQL language shall not contain a <non-parenthesized value expression primary> that simply contains an <md-axis variable>.

2) Insert after the last CR: Without Feature A010, "Multi-dimensional array support", conforming SQL language shall not contain a <collection value constructor> that simply contains an <md-array value constructor>.

3) Insert after the last CR: Without Feature A010, "Multi-dimensional array support", conforming SQL language shall not contain a <non-parenthesized value expression primary> that simply contains an <md-array aggregation expression>.

4) Insert after the last CR: Without Feature A010, "Multi-dimensional array support", conforming SQL language shall not contain a <non-parenthesized value expression primary> that simply contains an <md-array element reference>.

6.3 <md-array subset>

Function

Returns a subset of an MD-array.

Format

```
<md-array subset> ::=
  <md-array value expression>
      <left bracket or trigraph> <md-axis subset list> <right bracket or trigraph>

<md-axis subset list> ::=
    <md-axis subset list named>
  | <md-axis subset list positional>
  | <md-array subset extent>

<md-axis subset list named> ::=
  <md-axis subset named> [ { <comma> <md-axis subset named> }... ]

<md-axis subset list positional> ::=
  <md-axis subset positional> [ { <comma> <md-axis subset positional> }... ]

<md-axis subset named> ::=
    <md-axis limits named>
  | <md-axis slice named>

<md-axis subset positional> ::=
    <md-axis limits positional>
  | <md-axis slice positional>

<md-axis limits named> ::=
  <md-axis name> <left paren> <md-interval expression> <right paren>

<md-axis limits positional> ::=
  <md-interval expression>

<md-interval expression> ::=
  <md-axis lower limit expression> <colon> <md-axis upper limit expression>

<md-axis lower limit expression> ::=
  <md-axis limit expression>

<md-axis upper limit expression> ::=
  <md-axis limit expression>

<md-axis limit expression> ::=
    <numeric value expression>
  | <asterisk>

<md-axis slice named> ::=
  <md-axis name> <left paren> <md-axis slice positional> <right paren>

<md-axis slice positional> ::=
  <numeric value expression>

<md-array subset extent> ::=
  MDEXTENT <left paren> <md-array value expression> <right paren>
```

Syntax Rules

1) Let *AS* be the <md-array subset>, let *AVE* be the <md-array value expression> immediately contained in *AS*, and let *ASL* be the <md-axis subset list> immediately contained in *AS*.

2) Let *MDT* be the declared type of *AVE*.

3) The Syntax Rules of Subclause 9.16, "MD-array subset", are applied with *MDT* as *MDARRAY TYPE* and *ASL* as *MDAXIS SUBSET LIST*; let *TMD* be the *TARGET MAXIMUM MDEXTENT* and let *EQ* be the *EQUIVALENCE* returned from the application of those Syntax Rules.

   Case:

   a) If *EQ* is not a zero-length character string, then <md-array subset> is equivalent to `AVE EQ`.

   b) Otherwise:

      i) Let *EDT* be the element type of *AVE*.

      ii) The declared type of *AS* is an MD-array type of element type *EDT* and maximum MD-extent *TMD*.

Access Rules

No additional Access Rules.

General Rules

1) Let *AS* be the <md-array subset>, let *AVE* be the <md-array value expression> immediately contained in *AS*, and let *ASL* be the <md-axis subset list> immediately contained in *AS*. Let *AV* be the value of *AVE*.

2) If *AV* is the null value, then the result of *AS* is the null value and no further General Rules of this Subclause are applied.

3) The General Rules of Subclause 9.16, "MD-array subset", are applied with *AV* as *MDARRAY VALUE* and *ASL* as *MDAXIS SUBSET LIST*; let *SMD* be the *SOURCE MDEXTENT*, let *TMD* be the *TARGET MDEXTENT*, and let *SLICEDAXES* be the *SLICED MDAXES* returned from the application of those General Rules.

4) Let *D* be the MD-extent of *AV*, let *d* be the number of MD-axes in *D*, and let *t* be the number of MD-axes in *TMD*.

5) If *SMD* is not within *D*, then an exception condition is raised: *data exception — MD-array subset not within MD-extent (2203L)*.

6) Let *TV* be the MD-array value with element type *EDT* and MD-extent *TMD* constructed in the following way. For each coordinate *R* within *SMD*:

   a) Let RP_i be the *i*-th element of *R*, 1 (one) ≤ *i* ≤ *d*.

   b) Let *j* be 1 (one).

   c) For all *k*, 1 (one) ≤ *k* ≤ *d*, such that *k* is not an element of *SLICEDAXES*:

      i) Let TP_j be RP_k.

      ii) *j* is increased by 1 (one).

d) Let *Q* be the coordinate denoted by [TP_1, ..., TP_t].

NOTE 1 — There are *t* non-*SLICEDAXES*.

e) The element in *TV* at coordinate *Q* is the element in *AV* at coordinate *R*.

7) *TV* is the value of *AS*.

Conformance Rules

1) Without Feature A010, "Multi-dimensional array support", conforming SQL language shall not contain an <md-array subset>.

6.4 <identifier chain>

This Subclause modifies Subclause 6.6, "<identifier chain>", in ISO/IEC 9075-2.

Function

Disambiguate a period-separated chain of identifiers.

Format

No additional Format items.

Syntax Rules

1) Insert after SR 9)a)ii)1)C): The scope of an MD-axis variable equivalent to I_1.

2) Insert after SR 9)a)ii)2)B): If *IPST* is an <md-array aggregation expression> or an <md-array value constructor by iteration>, then let *SV* be the MD-axis variable equivalent to I_1. PIC_1 is the basis of *IC*, the basis length is 1 (one), the basis scope is the scope of *SP*, and the basis referent is *SV*.

Access Rules

No additional Access Rules.

General Rules

1) Insert after GR 4): If *BIC* is an MD-axis variable, then *BIC* references the MD-axis *SV* of a given execution of the <md-array aggregation expression> or an <md-array value constructor by iteration> whose <md-extent> contains *SV*.

Conformance Rules

No additional Conformance Rules.

6.5 <md-array aggregation expression>

Function

Aggregate MD-array values into a single value.

Format

```
<md-array aggregation expression> ::=
  MDAGGREGATE <md-array aggregation operator>
      OVER <md-extent alternative>
      USING <value expression primary>
      [ WHERE <search condition> ]

<md-array aggregation operator> ::=
    <plus sign> | AND | OR | MAX | MIN
```

Syntax Rules

1) The declared type of <md-array aggregation expression> *AAE* is the declared type of the immediately contained <value expression primary> *VE*. Let *op* be the immediately contained <md-array aggregation operator>.

   Case:

   a) If *op* is <plus sign>, then the declared type of *VE* shall be numeric.

   b) If *op* is AND or OR, then the declared type of *VE* shall be Boolean.

   c) Otherwise, *VE* is an operand of an ordering operation.

      i) The Syntax and Conformance Rules of Subclause 9.11, "Ordering operations", are applied.

      ii) The declared type of *VE* shall not be row-ordered.

2) If the <md-extent alternative> immediately contained in *AAE* is an <md-array extent> *EA*, then:

   a) Let *AVE1* be the <md-array value expression> immediately contained in *EA* and let *e* be the MD-dimension of *AVE1*.

   b) Case:

      i) If <search condition> *SC* is immediately contained in *AAE*, then *AEE* is equivalent to

```
MDAGGREGATE op
  OVER [ MDAXIS_NAME(AVE1, 1) ( MDAXIS_LOW(AVE1, 1) : MDAXIS_HIGH(AVE1, 1)
),
         ...,
         MDAXIS_NAME(AVE1, e) ( MDAXIS_LOW(AVE1, e) : MDAXIS_HIGH(AVE1, e)
)
       ]
USING  VE
WHERE  VE IS NOT NULL AND SC
```

      ii) Otherwise, *AEE* is equivalent to

```
MDAGGREGATE op
  OVER [ MDAXIS_NAME(AVE1, 1) ( MDAXIS_LOW(AVE1, 1) : MDAXIS_HIGH(AVE1, 1)
),
```

```
            ...,
            MDAXIS_NAME(AVE1, e) ( MDAXIS_LOW(AVE1, e) : MDAXIS_HIGH(AVE1, e)
)
        ]
USING   VE
WHERE   VE IS NOT NULL
```

3) Let *MD* be the maximum MD-extent specified by <md-extent> and let *d* be the number of MD-axes in *MD*. For 1 (one) ≤ *i* ≤ *d*, let AV_i be an <md-axis variable> whose name is equivalent to the name of the i-th MD-axis in *MD*. The scope of AV_i is *VE* and the <search condition> immediately contained in <md-array aggregation expression>, if specified.

Access Rules

None.

General Rules

1) Let *D* be the MD-extent specified by <md-extent>, let *d* be the number of MD-axes in *D*, and let N_i, LO_i, and HI_i, be the name, lower limit, and upper limit, respectively, of the *i*-th MD-axis in the maximum MD-extent of *D*, 1 (one) ≤ *i* ≤ *d*.

2) Let *S* be the union of all coordinates [P_1, ..., P_d], such that $LO_1 \le P_1 \le HI_1$, ..., $LO_d \le P_d \le HI_d$, (that is, *S* is the Cartesian product of all MD-intervals in *D*).

3) Let *op* be <md-array aggregation operator>. Let *AGG* be

   Case:

   a) If *op* is <plus sign>, then 0 (zero).

   b) If *op* is MAX, then an implementation-defined (IL067) minimum value of a type comparable to the <value expression primary>.

   c) If *op* is MIN, then an implementation-defined (IL068) maximum value of a type comparable to the <value expression primary>.

   d) If *op* is AND, then *True*.

   e) If *op* is OR, then *False*.

4) Let *NULLRES* be *True*.

5) For every coordinate [P_1, ..., P_d] in *S*,

   a) Let *BV* be *True*.

   b) If <search condition> *BVE* is specified, then the value of any AV_i contained in *BVE*, 1 (one) ≤ *i* ≤ *d*, is equal to P_i. Let *BV* then be the result of evaluating *BVE*.

   c) Case:

      i) If *BV* is *True*, then the value of any AV_i contained in *VE*, 1 (one) ≤ *i* ≤ *d*, is equal to P_i. Let *NULLRES* be *False*. Let the value of *VE* be *TEMP*.

         Case:

         1) If *op* is MIN, then let *AGG* be the lesser of *AGG* and *TEMP*.

2) If *op* is MAX, then let *AGG* be the greater of *AGG* and *TEMP*.

3) If *op* is AND or OR, then let *AGG* be the value of *AGG op TEMP*.

4) Otherwise, *op* is <plus sign>; let *TP* be the declared type of <md-array aggregation expression>.

Case:

A) If *TP* is exact numeric and the mathematical result of *AGG op TEMP* is not exactly representable with the precision and scale of *TP*, then an exception condition is raised: *data exception — numeric value out of range (22003)*.

B) If *TP* is approximate numeric and the exponent of the approximate mathematical result of *AGG op TEMP* is not within the implementation-defined (IL202) exponent range for *TP*, then an exception condition is raised: *data exception — numeric value out of range (22003)*.

C) If *TP* is the decimal floating point type, then the result is an implementation-defined (IV240) value *IV* of *TP* such that no other value of *TP* is strictly between *IV* and the mathematical result of *AGG op TEMP*. If the exponent of *IV* is not within the implementation-defined (IL202) exponent range for *TP*, then an exception condition is raised: *data exception — numeric value out of range (22003)*.

D) Otherwise, let *AGG* be the value of *AGG op TEMP*.

ii) Otherwise, *VE* is not evaluated for this coordinate.

6) Case:

a) If *NULLRES* is *False* then the result of <md-array aggregation expression> is *AGG*.

b) Otherwise, the result of <md-array aggregation expression> is the null value.

Conformance Rules

1) Without Feature A010, "Multi-dimensional array support", conforming SQL language shall not contain <md-array aggregation expression>.

6.6 <case expression>

This Subclause modifies Subclause 6.12, "<case expression>", in ISO/IEC 9075-2.

Function

Specify a conditional value.

Format

```
<searched when clause> ::=
    !! All alternatives from ISO/IEC 9075-2
  | WHEN <md-array boolean expression> THEN <result>
```

Syntax Rules

1) Insert after SR 2): If a <case specification> specifies a <searched case>, then:

   a) Let N be the number of <searched when clause>s.

   b) For each i between 1 (one) and N, let SWC_i be the i-th <searched when clause>, let SC_i be the immediately contained <search condition> or <md-array boolean expression> of SWC_i, and let R_i be the <result> immediately contained in SWC_i.

   c) Case:

      i) If <else clause> is specified, then let M be $N + 1$. Let R_M be the <result> immediately contained in the <else clause>.

      ii) Otherwise, let M be N.

   d) Either every SC_i, 1 (one) $\leq i \leq N$, shall be an <md-array boolean expression>, or no SC_i, 1 (one) $\leq i \leq N$, shall be an <md-array boolean expression>.

   e) If every SC_i is an <md-array boolean expression> MBE_i, then let MDT_i be the declared type of MBE_i and let d_i be the MD-dimension of MDT_i. For all j and k, $j \neq k$ and 1 (one) $\leq j, k \leq N$, d_j shall be equal to d_k. Let MN_i be the name of the i-th MD-axis in the maximum MD-extent of MDT_1, 1 (one) $\leq i \leq d_1$.

      i) <case specification> shall not generally contain a <routine invocation> whose subject routine is an SQL-invoked routine that is possibly non-deterministic or that possibly modifies SQL-data.

      ii) <case specification> shall not generally contain a <table primary> that contains a <data change delta table>.

      iii) For all j, 1 (one) $\leq j \leq M$:

         Case:

         1) If R_j is of MD-array type, then let MDT be the declared type of R_j, and let e be the MD-dimension of MDT. e shall be equal to d_1. Let RA_j be R_j.

         2) Otherwise, let e be d_1. RA_j is equivalent to

            MDARRAY [MN_1 (MDAXIS_LOW (SC_1, 1) : MDAXIS_HIGH (SC_1, 1)), ...,

```
                    MN_e ( MDAXIS_LOW (SC_1, e) : MDAXIS_HIGH (SC_1, e) ) ]
ELEMENTS R_j
```

Access Rules

None.

General Rules

1) Insert before GR 2)a): If the value of every SC_i, 1 (one) ≤ i ≤ N, is an MD-array value, then:

   a) Let D_i be the MD-extent of the value of SC_i. For all j and k, $j \neq k$ and 1 (one) ≤ j, k ≤ N, if D_j is not equal to D_k, then an exception condition is raised: *data exception — MD-array operands with non-matching MD-extents (2203R)*.

   b) For all j, 1 (one) ≤ j ≤ M, let E_j be the MD-extent of the value of RA_j. If E_j is not equal to D_1, then an exception condition is raised: *data exception — MD-array operands with non-matching MD-extents (2203R)*.

   c) If <else clause> is specified, then let *CEEC* be equivalent to ELSE RA_M[N_1, ..., N_d]; otherwise, let *CEEC* be a character string of length 0 (zero).

   d) Let d be d_1, and let N_i, LO_i, and HI_i, be the name, lower limit, and upper limit, respectively, of the i-th MD-axis in the MD-extent of the value of SC_1, 1 (one) ≤ i ≤ d.

   e) The value of the <case specification> is the value of:

```
MDARRAY [ N_1 ( LO_1 : HI_1 ), ...,
          N_d ( LO_d : HI_d ) ]
ELEMENTS
CASE WHEN SC_1[ N_1, ..., N_d ] THEN RA_1[ N_1, ..., N_d ]
     ...
     WHEN SC_N[ N_1, ..., N_d ] THEN RA_N[ N_1, ..., N_d ]
     CEEC END
```

Conformance Rules

1) Insert after the last CR: Without Feature A010, "Multi-dimensional array support", in conforming SQL language, a <case expression> shall not support operands of MD-array type.

2) Insert after the last CR: Without Feature A011, "MD-array BOOLEAN element type", conforming SQL language shall not contain a <searched when clause> that immediately contains an <md-array boolean expression>.

6.7 <cast specification>

This Subclause modifies Subclause 6.13, "<cast specification>", in ISO/IEC 9075-2.

Function

Specify a data conversion.

Format

```
<cast target> ::=
    !! All alternatives from ISO/IEC 9075-2
  | <md-array cast>
  | <md-array base type cast>
  | <md-array maximum md-extent cast>

<md-array cast> ::=
  <data type> MDARRAY <md-array axis names>

<md-array base type cast> ::=
  <data type> MDARRAY

<md-array maximum md-extent cast> ::=
  MDARRAY <maximum md-extent cast alternative>

<maximum md-extent cast alternative> ::=
    <maximum md-extent>
  | <md-array axis names>

<md-axis explicit name list> ::=
  <left bracket or trigraph> <md-axis name>
      [ { <comma> <md-axis name> }... ] <right bracket or trigraph>

<md-array axis names> ::=
  MDAXIS_NAMES <left paren> <md-array value expression> <right paren>
```

Syntax Rules

1) Insert before SR 1ja): If <cast target> immediately contains an <md-array cast>, <md-array base type cast>, <md-array maximum md-extent cast>, or <data type> *TD* that is an <md-array type>, then:

   a) Neither <cast operand> nor <cast target> shall generally contain a <routine invocation> whose subject routine is an SQL-invoked routine that is possibly non-deterministic or that possibly modifies SQL-data.

   b) Neither <cast operand> nor <cast target> shall generally contain a <table primary> that contains a <data change delta table>.

   c) The <cast operand> shall be an <md-array value expression>. Let *AVE* be the <cast operand>. Let *MDT* be the declared type of *AVE*, let *EDT* be the element type of *MDT*, let *d* be the MD-dimension of *MDT*, and let MN_i be the name of the *i*-th MD-axis in the maximum MD-extent of *MDT*, 1 (one) ≤ *i* ≤ *d*.

      Case:

      i) If an <md-array cast> *AC* is specified, then let *TEDT* be the data type identified by the <data type> immediately contained in *AC* and let *ANC* be the <md-array axis names>

immediately contained in *AC*. *TEDT* shall not be a collection-containing type. The <cast specification> is equivalent to

```
CAST(CAST(AVE AS TEDT MDARRAY) AS ANC)
```

ii) If *TD* is an <md-array type>, then let *TEDT* be the <data type> immediately contained in *TD*, and let *ANC* be the <maximum md-extent> immediately contained in *TD*. *TEDT* shall not be a collection-containing type. The <cast specification> is equivalent to

```
CAST(CAST(AVE AS TEDT MDARRAY) AS MDARRAY ANC)
```

iii) If an <md-array base type cast> *ABTC* is specified, then let *TEDT* be the data type identified by the <data type> immediately contained in *ABTC*. The <cast specification> is equivalent to

```
MDARRAY [ MN1 ( MDAXIS_LOW (AVE, 1) : MDAXIS_HIGH (AVE, 1) ), ...,
          MNd ( MDAXIS_LOW (AVE, d) : MDAXIS_HIGH (AVE, d) ) ]
ELEMENTS CAST(AVE[ MN1, ..., MNd ] AS TEDT)
```

iv) Otherwise, an <md-array maximum md-extent cast> *MMC* is specified. Let *MMCA* be the <maximum md-extent cast alternative> immediately contained in *MMC*.

Case:

1) If *MMCA* is a <maximum md-extent> *TMD*, then the MD-dimension of *TMD* shall be equal to *d*. *TMD* is the *maximum MD-extent* of *TD*.

2) Otherwise, let *AVE1* be the <md-array value expression> immediately contained in the <md-array axis names> immediately contained in *MMCA*, and let *e* be the MD-dimension of *AVE1*. *e* shall be equal to *d*. The <cast specification> is equivalent to:

```
CAST(AVE AS MDARRAY [ MDAXIS_NAME(AVE1, 1) ( MDAXIS_LOW(AVE1, 1) :
MDAXIS_HIGH(AVE1, 1) ),
                      ...,
                      MDAXIS_NAME(AVE1, e) ( MDAXIS_LOW(AVE1, e) :
MDAXIS_HIGH(AVE1, e) ) ] )
```

Access Rules

No additional Access Rules.

General Rules

1) Insert before GR 7): If <cast target> simply contains a <maximum md-extent>, then:

a) Let LO_i and HI_i, 1 (one) $\leq i \leq d$, be the lower limit and upper limit, respectively, of the *i*-th MD-axis in the MD-extent of the value of *AVE*. Let TMN_i, be the name of the *i*-th MD-axis in *TMD*.

b) Let *D* be the MD-extent denoted by [TMN_1 (LO_1 : HI_1), ..., TMN_d (LO_d : HI_d)].

Case:

i) If *D* is not strictly within *TMD*, then an exception condition is raised: *data exception — MD-array source MD-extent not strictly within maximum target MD-extent (2203Q)*.

ii) Otherwise, *TV* is the value of *AVE* with MD-extent *D*.

Conformance Rules

1) Insert after the last CR: Without Feature A010, "Multi-dimensional array support", conforming SQL language shall not contain a <cast target> that immediately contains an <md-array cast>, an <md-array base type cast>, or an <md-array maximum md-extent cast>.

2) Insert after the last CR: Without Feature A010, "Multi-dimensional array support", in conforming SQL language, <cast specification> shall not support <cast operand>s of MD-array type.

6.8 <value expression>

This Subclause modifies Subclause 6.29, "<value expression>", in ISO/IEC 9075-2.

Function

Specify a value.

Format

```
<collection value expression> ::=
    !! All alternatives from ISO/IEC 9075-2
  | <md-array value expression>
```

Syntax Rules

1) Augment SR 5) by adding "<md-array value expression>" to the list of BNF non-terminals.

Access Rules

No additional Access Rules.

General Rules

1) Augment GR 4) by adding "<md-array value expression>" to the list of BNF non-terminals.

Conformance Rules

1) Insert after the last CR: Without Feature A010, "Multi-dimensional array support", conforming SQL language shall not contain a <collection value expression> that immediately contains an <md-array value expression>.

6.9 <numeric value function>

This Subclause modifies Subclause 6.31, "<numeric value function>", in ISO/IEC 9075-2.

Function

Specify a function yielding a value of type numeric.

Format

```
<numeric value function> ::=
    !! All alternatives from ISO/IEC 9075-2
  | <md-array dimension>
  | <md-array axis index function>
  | <md-array lower axis limit>
  | <md-array upper axis limit>
  | <md-array aggregation function>

<md-array dimension> ::=
  MDDIMENSION <left paren> <md-array value expression> <right paren>

<md-array axis index function> ::=
  MDAXIS_INDEX <left paren>
      <md-array value expression> <comma> <md-axis name>
      <right paren>

<md-array lower axis limit> ::=
  MDAXIS_LOW <left paren> <md-array md-axis> <right paren>

<md-array upper axis limit> ::=
  MDAXIS_HIGH <left paren> <md-array md-axis> <right paren>

<md-array aggregation function> ::=
  <md-array aggregation function name>
      <left paren> <md-array value expression> <right paren>

<md-array aggregation function name> ::=
    MDALL
  | MDANY
  | MDAVG
  | MDCOUNT
  | MDCOUNT_FALSE
  | MDCOUNT_TRUE
  | MDCOUNT_UNKNOWN
  | MDMAX
  | MDMIN
  | MDSUM
```


Syntax Rules

1) Insert after the last SR: If <md-array dimension>, <md-array axis index function>, <md-array lower axis limit>, <md-array upper axis limit>, or <md-array aggregation function> is specified, then let *AVE* be the simply contained <md-array value expression>. Let *MDT* be the declared type of *AVE*, let *EDT* be the element type of *MDT*, let *d* be the MD-dimension of *MDT*, and let MN_i be the name of the *i*-th MD-axis in the maximum MD-extent of *MDT*, 1 (one) $\leq i \leq d$.

2) Insert after the last SR: If <md-array dimension> is specified, then the declared type of the result is an implementation-defined (IV241) exact numeric type with scale 0 (zero).

3) Insert after the last SR: If <md-array axis index function> *AIF* is specified, then the declared type of the result is an implementation-defined (IV242) exact numeric type with scale 0 (zero). The <md-axis name> immediately contained in *AIF* shall be equivalent to MN_{idx} for some *idx*, 1 (one) ≤ *idx* ≤ *d*.

4) Insert after the last SR: If <md-array lower axis limit> or <md-array upper axis limit> is specified, then the declared type of the result is an implementation-defined (IV243) exact numeric type with scale 0 (zero). Let *AD* be the index specified by <md-array md-axis>.

5) Insert after the last SR: If <md-array aggregation function> *MAF* is specified, then *MAF* shall not generally contain a <routine invocation> whose subject routine is an SQL-invoked routine that is possibly non-deterministic or that possibly modifies SQL-data, and *MAF* shall not generally contain a <table primary> that contains a <data change delta table>.

Case:

a) If <md-array aggregation function name> is one of MDSUM, MDMAX, MDMIN, MDALL, or MDANY, then let *op* be <plus sign> for MDSUM, MAX for MDMAX, MIN for MDMIN, AND for MDALL, and OR for MDANY. <md-array aggregation function> is equivalent to:

```
MDAGGREGATE op
OVER [ MN1 ( MDAXIS_LOW (AVE, 1) : MDAXIS_HIGH (AVE, 1) ),
       ...,
       MNd ( MDAXIS_LOW (AVE, d) : MDAXIS_HIGH (AVE, d) ) ]
USING AVE[ MN1, ..., MNd ]
WHERE AVE[ MN1, ..., MNd ] IS NOT NULL
```

b) If <md-array aggregation function name> is MDCOUNT_TRUE, then <md-array aggregation function> is equivalent to:

```
MDAGGREGATE <plus sign>
OVER [ MN1 ( MDAXIS_LOW (AVE, 1) : MDAXIS_HIGH (AVE, 1) ),
       ...,
       MNd ( MDAXIS_LOW (AVE, d) : MDAXIS_HIGH (AVE, d) ) ]
USING CASE WHEN AVE[ MN1, ..., MNd ] THEN 1 ELSE 0 END
WHERE AVE[ MN1, ..., MNd ] IS NOT NULL
```

c) If <md-array aggregation function name> is MDCOUNT_FALSE, then <md-array aggregation function> is equivalent to:

```
MDCOUNT_TRUE( AVE IS FALSE )
```

d) If <md-array aggregation function name> is MDCOUNT_UNKNOWN, then <md-array aggregation function> is equivalent to:

```
MDCOUNT_TRUE( AVE IS UNKNOWN )
```

e) If <md-array aggregation function name> is MDCOUNT, then <md-array aggregation function> is equivalent to:

```
MDAGGREGATE <plus sign>
OVER [ MN1 ( MDAXIS_LOW (AVE, 1) : MDAXIS_HIGH (AVE, 1) ),
       ...,
       MNd ( MDAXIS_LOW (AVE, d) : MDAXIS_HIGH (AVE, d) ) ]
USING 1
WHERE AVE[ MN1, ..., MNd ] IS NOT NULL
```

f) Otherwise, <md-array aggregation function name> is MDAVG and <md-array aggregation function> is equivalent to:

```
CASE WHEN MDCOUNT ( AVE ) = 0 THEN NULL
     ELSE MDSUM ( AVE ) / MDCOUNT ( AVE ) END
```

Access Rules

No additional Access Rules.

General Rules

1) Insert after the last GR: The result of <md-array dimension> is *d*.

2) Insert after the last GR: The result of <md-array axis index function> is *idx*.

3) Insert after the last GR: If <md-array lower axis limit> or <md-array upper axis limit> is specified, then:

   a) Let *VAVE* be the value of *AVE*.

   b) Case:

      i) If *VAVE* is the null value, then the result is the null value.

      ii) Otherwise, let LO_i and HI_i be the lower and upper limits of the *i*-th MD-axis of the MD-extent of *VAVE*.

      Case:

      1) If <md-array lower axis limit> is specified, then the result is LO_{AD}.

      2) Otherwise, the result is HI_{AD}.

Conformance Rules

1) Insert after the last CR: Without Feature A010, "Multi-dimensional array support", conforming SQL language shall not contain <md-array dimension>, <md-array lower axis limit>, <md-array upper axis limit>, or <md-array axis index function>.

2) Insert after the last CR: Without Feature A010, "Multi-dimensional array support", conforming SQL language shall not contain <md-array aggregation function>.

3) Insert after the last CR: Without Feature A011, "MD-array BOOLEAN element type", conforming SQL language shall not contain an <md-array aggregation function name> that is MDALL, MDANY, MDCOUNT_FALSE, MDCOUNT_TRUE, or MDCOUNT_UNKNOWN.

6.10 <string value function>

This Subclause modifies Subclause 6.33, "<string value function>", in ISO/IEC 9075-2.

Function

Specify a function yielding a value of type character string or binary string.

Format

```
<string value function> ::=
    !! All alternatives from ISO/IEC 9075-2
  | <md-array encode function>

<character value function> ::=
    !! All alternatives from ISO/IEC 9075-2
  | <md-array axis name function>

<md-array axis name function> ::=
  MDAXIS_NAME <left paren>
      <md-array value expression> <comma> <numeric value expression>
      <right paren>
```

Syntax Rules

1) Augment SR 1) by adding "<md-array encode function>" to the list of BNF non-terminals.

2) Augment SR 2) by adding "<md-array axis name function>" to the list of BNF non-terminals.

3) Insert after the last SR: If <md-array axis name function> *ANF* is specified, then:

   a) Let *MDT* be the declared type of the <md-array value expression> immediately contained in *ANF*, let *MD* be the maximum MD-extent of *MDT*, let *d* be the number of MD-axes in *MDT*, and let MN_i be the name of the *i*-th MD-axis in *MD*, 1 (one) ≤ *i* ≤ *d*.

   b) Let *NVE* be the <numeric value expression> immediately contained in *ANF*. The declared type of *NVE* shall be an exact numeric type with scale 0 (zero).

   c) Let *VL* be the implementation-defined (IL006) maximum length of variable-length character strings. The declared type of the result is variable-length character string type with maximum length equal to *VL*.

Access Rules

No additional Access Rules.

General Rules

1) Augment GR 1) by adding "<md-array encode function>" to the list of BNF non-terminals.

2) Augment GR 2) by adding "<md-array axis name function>" to the list of BNF non-terminals.

3) Insert after the last GR: The result of <md-array axis name function> is determined as follows:

   a) Let *NV* be the value of *NVE*.

b) If *NV* is the null value, then the result is the null value and no further General Rules of this Subclause are applied.

c) If *NV* is less than 1 (one) or greater than *d*, then an exception condition is raised: *data exception — MD-array invalid MD-axis (2203T)*.

d) Let *AVE* be the value of the <md-array value expression>.

e) If *AVE* is the null value, then the result is the null value and no further General Rules of this Subclause are applied.

f) The result is MN_{NV}.

Conformance Rules

1) Insert after the last CR: Without Feature A010, "Multi-dimensional array support", conforming SQL language shall not contain <md-array axis name function>.

2) Insert after the last CR: Without Feature A010, "Multi-dimensional array support", conforming SQL language shall not contain a <string value function> that is an <md-array encode function>.

6.11 <md-array encode function>

Function

Convert an array to format-specific encoded string.

Format

```
<md-array encode function> ::=
  MDENCODE <left paren>
      <md-array value expression> <comma> <format identifier>
      [ <md-array encode returning clause> ]
  <right paren>

<format identifier> ::=
  <character string literal>

<md-array encode returning clause> ::=
  RETURNING <data type>
```

Syntax Rules

1) Let *AEF* be the <md-array encode function> and let *FID* be the value of <format identifier> immediately contained in *AEF*. *FID* shall be a valid media type as specified in RFC 2046.

2) If <md-array encode returning clause> is not specified, then

   Case:

   a) If *FID* is equivalent to 'application/json', then an implementation-defined (ID225) character string type is implicit.

   b) Otherwise, an implementation-defined (ID226) string type is implicit.

3) The <data type> *DT* immediately contained in the explicit or implicit <md-array encode returning clause> shall be a <predefined type> that identifies a character string data type or binary string data type.

4) The declared type of *AEF* is the data type specified by *DT*.

Access Rules

None.

General Rules

1) Let *AVE* be the value of the <md-array value expression> immediately contained in *AEF*.

2) If *AVE* is the null value, then the result of *AEF* is the null value and no further General Rules of this Subclause are applied.

3) Let *ET* be the element type of the declared type of *AVE*.

   Case:

   a) If *FID* is equivalent to 'application/json', then

i) The function *F* transforming an MD-array *A* to JSON text is defined recursively as follows.

Let *N* be the value of MDAXIS_NAME(*A*, 1), let *LO* be the value of MDAXIS_LOW(*A*, 1), and let *HI* be the value of MDAXIS_HIGH(*A*, 1).

Case:

1) If MDDIMENSION(*A*) is equal to 1 (one), then

Case:

A) If *ET* is a row type, then let *DET* be the degree of *ET* and let FN_j be the name of the *j*-th field of *ET*, 1 (one) ≤ *j* ≤ *DET*. *F*(*A*) is the result of

```
JSON_ARRAY(
  JSON_OBJECT( FN1 : A[LO].FN1, ..., FNDET : A[LO].FNDET ),
  ...,
  JSON_OBJECT( FN1 : A[HI].FN1, ..., FNDET : A[HI].FNDET ),
)
```

B) Otherwise, *F*(*A*) is the result of

```
JSON_ARRAY( A[LO], ..., A[HI] )
```

2) Otherwise, let G_i be *F*(*A*[*N*(*i*)]), *LO* ≤ *i* ≤ *HI*. *F*(*A*) is the result of

```
JSON_ARRAY(GLO, ..., GHI)
```

ii) The result of *AEF* is JSON_OBJECT('data' VALUE *F*(*AVE*) FORMAT JSON).

b) Otherwise, the result of *AEF* is implementation-defined (IV244). If an encoding error occurs, then an exception condition is raised: *data exception — MD-array encoding error (2203Z)*.

Conformance Rules

1) Without Feature A014, "MDA encode/decode functions media type: application/json", conforming SQL language shall not contain <md-array encode function>.

2) Without Feature A015, "MDA encode/decode functions with implementation-defined behavior", conforming SQL language shall not contain an <md-array encode function> that contains a <format identifier> that is not equivalent to 'application/json'.

6.12 <md-array value expression>

Function

Specify an MD-array value.

Format

```
<md-array value expression> ::=
    <md-array boolean expression>
  | <md-array numeric expression>
  | <md-array primary>

<md-array boolean expression> ::=
    <md-array boolean term>
  | <md-array boolean expression left>
  | <md-array boolean expression right>
  | <md-array boolean expression both>

<md-array boolean expression left> ::=
  <md-array boolean expression> OR <boolean term>

<md-array boolean expression right> ::=
  <boolean value expression> OR <md-array boolean term>

<md-array boolean expression both> ::=
  <md-array boolean expression> OR <md-array boolean term>

<md-array boolean term> ::=
    <md-array boolean factor>
  | <md-array boolean term left>
  | <md-array boolean term right>
  | <md-array boolean term both>

<md-array boolean term left> ::=
  <md-array boolean term> AND <boolean factor>

<md-array boolean term right> ::=
  <boolean term> AND <md-array boolean factor>

<md-array boolean term both> ::=
  <md-array boolean term> AND <md-array boolean factor>

<md-array boolean factor> ::=
  [ NOT ] <md-array boolean test>

<md-array boolean test> ::=
  <md-array comparison expression> [ IS [ NOT ] <truth value> ]

<md-array comparison expression> ::=
    <md-array boolean primary>
  | <md-array comparison expression left>
  | <md-array comparison expression right>
  | <md-array comparison expression both>

<md-array comparison expression left> ::=
  <md-array numeric expression> <comp op> <numeric value expression>

<md-array comparison expression right> ::=
  <numeric value expression> <comp op> <md-array numeric expression>
```

```
<md-array comparison expression both> ::=
  <md-array numeric expression> <comp op> <md-array numeric expression>

<md-array boolean primary> ::=
    <md-array primary>
  | <parenthesized md-array boolean expression>

<parenthesized md-array boolean expression> ::=
  <left paren> <md-array boolean expression> <right paren>

<md-array numeric expression> ::=
    <md-array numeric term>
  | <md-array numeric expression left>
  | <md-array numeric expression right>
  | <md-array numeric expression both>

<md-array numeric expression left> ::=
    <md-array numeric expression> <plus sign> <term>
  | <md-array numeric expression> <minus sign> <term>

<md-array numeric expression right> ::=
    <numeric value expression> <plus sign> <md-array numeric term>
  | <numeric value expression> <minus sign> <md-array numeric term>

<md-array numeric expression both> ::=
    <md-array numeric expression> <plus sign> <md-array numeric term>
  | <md-array numeric expression> <minus sign> <md-array numeric term>

<md-array numeric term> ::=
    <md-array numeric factor>
  | <md-array numeric term left>
  | <md-array numeric term right>
  | <md-array numeric term both>

<md-array numeric term left> ::=
    <md-array numeric term> <asterisk> <factor>
  | <md-array numeric term> <solidus> <factor>

<md-array numeric term right> ::=
    <term> <asterisk> <md-array numeric factor>
  | <term> <solidus> <md-array numeric factor>

<md-array numeric term both> ::=
    <md-array numeric term> <asterisk> <md-array numeric factor>
  | <md-array numeric term> <solidus> <md-array numeric factor>

<md-array numeric factor> ::=
  [ <sign> ] <md-array numeric primary>

<md-array numeric primary> ::=
  <md-array primary>

<md-array field reference> ::=
  <md-array primary> <period> <field name>

<md-array primary> ::=
    <value expression primary>
  | <md-array subset>
  | <md-array value function>
  | <md-array field reference>
```

Syntax Rules

1) The declared type of <md-array value expression> is the declared type of the immediately contained <md-array boolean expression>, <md-array numeric expression>, or <md-array primary>.

2) The declared type of <md-array numeric primary> shall be an MD-array type whose element type is a numeric type.

3) The declared type of <md-array boolean primary> shall be an MD-array type whose element type is Boolean.

4) The declared type of <value expression primary> immediately contained in <md-array primary> shall be an MD-array type.

5) If <md-array boolean factor>, <md-array boolean test>, <md-array numeric factor>, or <md-array field reference> is contained in an <md-array value expression> *MAVE*, then:

   a) Let *OP1* and *OP2* be zero-length character strings.

   b) Case:

      i) If <md-array boolean factor> *E* is simply contained in *MAVE*, then let *AVE* be the <md-array boolean test> immediately contained in *E*. If NOT is immediately contained in *E*, then let *OP1* be `NOT`.

      ii) If <md-array boolean test> *E* is simply contained in *MAVE*, then let *AVE* be the <md-array comparison expression> immediately contained in *E*. If <truth value> *TV* is immediately contained in *E*, then

          Case:

          1) If NOT is immediately contained in *E*, then let *OP2* be `IS NOT` *TV*.

          2) Otherwise, let *OP2* be `IS` *TV*.

      iii) If <md-array numeric factor> *E* is simply contained in *MAVE*, then let *AVE* be the <md-array numeric primary> immediately contained in *E*. If <sign> *S* is immediately contained in *E*, then let *OP1* be *S*.

      iv) If <md-array field reference> *E* is simply contained in *MAVE*, then let *AVE* be the <md-array primary> immediately contained in *E*, let *EDT* be the declared element type of *MDT*, and let *FN* be the <field name> immediately contained in *E*.

          1) *EDT* shall be a row type.

          2) *FN* shall unambiguously reference a field of *EDT*.

          3) Let *OP2* be <period> *FN*.

   c) If *OP1* or *OP2* is a character string of length greater than 0 (zero), then:

      i) *E* shall not generally contain a <routine invocation> whose subject routine is an SQL-invoked routine that is possibly non-deterministic or that possibly modifies SQL-data.

      ii) *E* shall not generally contain a <table primary> that contains a <data change delta table>.

      iii) Let *MDT* be the declared type of *AVE*. *MDT* shall be an MD-array type.

      iv) Let *MD* be the maximum MD-extent of *MDT*, let *d* be the number of MD-axes in *MD*, and let MN_i be the name of the *i*-th MD-axis in *MD*, 1 (one) ≤ *i* ≤ *d*.

v) *E* is equivalent to

```
MDARRAY [ MN1 ( MDAXIS_LOW(AVE, 1) : MDAXIS_HIGH(AVE, 1) ),
          ...,
          MNd ( MDAXIS_LOW(AVE, d) : MDAXIS_HIGH(AVE, d) ) ]
  ELEMENTS ( OP1 AVE[MN1, ..., MNd] OP2 )
```

6) If <md-array boolean expression>, <md-array boolean term>, <md-array comparison expression>, <md-array numeric expression>, or <md-array numeric term> is simply contained in an <md-array value expression> *MAVE*, then:

a) Case:

i) If *MAVE* is an <md-array boolean expression> *BE*, and one of <md-array boolean expression left>, <md-array boolean expression right>, or <md-array boolean expression both> is immediately contained in *BE*, then:

1) Let *E* be the <md-array boolean expression left>, <md-array boolean expression right>, or <md-array boolean expression both> immediately contained in *BE*.

2) Let *V1* be the <md-array boolean expression> or <boolean value expression> immediately contained in *E* and let *V2* be the <md-array boolean term> or <boolean term> immediately contained in *E*.

3) Let *OP* be OR.

ii) If <md-array boolean term> *BE* is simply contained in *MAVE*, and one of <md-array boolean term left>, <md-array boolean term right>, or <md-array boolean term both> is immediately contained in *BE*, then:

1) Let *E* be the <md-array boolean term left>, <md-array boolean term right>, or <md-array boolean term both> immediately contained in *BE*.

2) Let *V1* be the <md-array boolean term> or <boolean term> immediately contained in *E* and let *V2* be the <md-array boolean factor> or <boolean factor> immediately contained in *E*.

3) Let *OP* be AND.

iii) If <md-array comparison expression> *BE* is simply contained in *MAVE*, and one of <md-array comparison expression left>, <md-array comparison expression right>, or <md-array comparison expression both> is immediately contained in *BE*, then:

1) Let *E* be the <md-array comparison expression left>, <md-array comparison expression right>, or <md-array comparison expression both> immediately contained in *BE*.

2) Let *V1* be the first <md-array numeric expression> or <numeric value expression> immediately contained in *E* and let *V2* be the second <md-array numeric expression> or <numeric value expression> immediately contained in *E*.

3) Let *OP* be the <comp op> immediately contained in *E*.

iv) If *MAVE* is an <md-array numeric expression> *BE* and one of <md-array numeric expression left>, <md-array numeric expression right>, or <md-array numeric expression both> is immediately contained in *BE*, then:

1) Let *E* be the <md-array numeric expression left>, <md-array numeric expression right>, or <md-array numeric expression both> immediately contained in *BE*.

2) Let *V1* be the <md-array numeric expression> or <numeric value expression> immediately contained in *E* and let *V2* be the <md-array numeric term> or <term> immediately contained in *E*.

3) Case:

A) If <plus sign> is immediately contained in *E*, then let *OP* be <plus sign>.

B) Otherwise, let *OP* be <minus sign>. If *V2* is a <factor> or an <md-array numeric factor> that immediately contains a <sign> that is a <minus sign> *FMS*, then there shall be a <separator> between *OP* and *FMS*.

v) If <md-array numeric term> *BE* is simply contained in *MAVE*, and one of <md-array numeric term left>, <md-array numeric term right>, or <md-array numeric term both> is immediately contained in *BE*, then:

1) Let *E* be the <md-array numeric term left>, <md-array numeric term right>, or <md-array numeric term both> immediately contained in *BE*.

2) Let *V1* be the <md-array numeric term> or <term> immediately contained in *E* and let *V2* be the <md-array numeric factor> or <factor> immediately contained in *E*.

3) Let *OP* be the <asterisk> or <solidus> immediately contained in *E*.

b) *BE* shall not generally contain a <routine invocation> whose subject routine is an SQL-invoked routine that is possibly non-deterministic or that possibly modifies SQL-data.

c) *BE* shall not generally contain a <table primary> that contains a <data change delta table>.

d) Let *T1* be the declared type of *V1* and let *T2* be the declared type of *V2*.

e) Case:

i) If *T1* is an MD-array type, then let *MD1* be the maximum MD-extent of *T1*, let *d1* be the number of MD-axes in *MD1*, let $MN1_i$ be the name of the *i*-th MD-axis in *MD1*, 1 (one) $\le i \le d1$. Let *d* be *d1*, let *MD* be *MD1*, let MN_i be $MN1_i$, let *AVE* be *V1*, and let *EXP1* be *V1*[MN_1, ..., MN_d].

ii) Otherwise, let *EXP1* be *V1*.

f) Case:

i) If *T2* is an MD-array type, then let *MD2* be the maximum MD-extent of *T2*, let *d2* be the number of MD-axes in *MD2*, let $MN2_i$ be the name of the *i*-th MD-axis in *MD2*, 1 (one) $\le i \le d2$. Let *d* be *d2*, let *MD* be *MD2*, let MN_i be $MN2_i$, let *AVE* be *V2*, and let *EXP2* be *V2*[MN_1, ..., MN_d].

ii) Otherwise, let *EXP2* be *V2*.

g) If both *T1* and *T2* are MD-array types, then *d1* shall be equal to *d2* and $MN1_i$ shall be equivalent to $MN2_i$, 1 (one) $\le i \le d$.

h) The declared type of *E* is an MD-array type with maximum MD-extent *MD*.

i) If only one of *T1* or *T2* is an MD-array type, then *E* is equivalent to

```
MDARRAY [ MN1( MDAXIS_LOW(AVE, 1) : MDAXIS_HIGH(AVE, 1) ),
          ...,
          MNd ( MDAXIS_LOW(AVE, d) : MDAXIS_HIGH(AVE, d) ) ]
  ELEMENTS ( EXP1 OP EXP2 )
```

Access Rules

No additional Access Rules.

General Rules

1) If <md-array comparison expression>, <md-array numeric expression>, <md-array numeric term>, <md-array boolean expression>, or <md-array boolean term> is simply contained in an <md-array value expression>, then let *LD* be the MD-extent of *V1* and let *RD* be the MD-extent of *V2*.

   Case:

   a) If *LD* is not equal to *RD*, then an exception condition is raised: *data exception — MD-array operands with non-matching MD-extents (2203R)*.

   b) Otherwise, the result of *E* is:

```
MDARRAY [ MN1 ( MDAXIS_LOW(AVE, 1) : MDAXIS_HIGH(AVE, 1) ),
          ...,
          MNd ( MDAXIS_LOW(AVE, d) : MDAXIS_HIGH(AVE, d) ) ]
ELEMENTS ( EXP1 OP EXP2 )
```

Conformance Rules

1) Without Feature A010, "Multi-dimensional array support", conforming SQL language shall not contain an <md-array value expression>.

2) Without Feature A012, "MD-array truth value tests", conforming SQL language shall not contain an <md-array boolean test> that contains a <truth value>.

3) Without Feature A011, "MD-array BOOLEAN element type", conforming SQL language shall not contain an <md-array boolean primary> that simply contains an <md-array primary>.

4) Without Feature A011, "MD-array BOOLEAN element type", conforming SQL language shall not contain an <md-array value expression> that simply contains an <md-array boolean expression>.

6.13 <md-array value function>

Function

Specify a function yielding a value of an MD-array type.

Format

```
<md-array value function> ::=
    <md-array shift function>
  | <md-array reshape function>
  | <md-array scale function>
  | <md-array absolute value expression>
  | <md-array natural logarithm>
  | <md-array common logarithm>
  | <md-array exponential function>
  | <md-array square root>
  | <md-array floor function>
  | <md-array ceiling function>
  | <md-array trigonometric function>
  | <md-array power function>
  | <md-array modulus expression>
  | <md-array concatenation>

<md-array reshape function> ::=
  MDRESHAPE <left paren>
      <md-array value expression> <comma> <md-axis limits list>
      <right paren>

<md-array shift function> ::=
  MDSHIFT <left paren>
      <md-array value expression> <comma>
      <left bracket or trigraph> <md-axis slice list> <right bracket or trigraph>
  <right paren>

<md-array scale function> ::=
  MDSCALE <left paren>
      <md-array value expression> <comma> <md-axis limits list>
      <right paren>

<md-axis limits list> ::=
    <left bracket or trigraph> <md-axis limits list positional> <right bracket or trigraph>
  | <left bracket or trigraph> <md-axis limits list named> <right bracket or trigraph>
  | <md-array extent>

<md-axis limits list positional> ::=
  <left bracket or trigraph>
      <md-axis limits positional> [ { <comma> <md-axis limits positional> }... ]
  <right bracket or trigraph>

<md-axis limits list named> ::=
  <left bracket or trigraph>
      <md-axis limits named> [ { <comma> <md-axis limits named> }... ]
  <right bracket or trigraph>

<md-array power function> ::=
    <md-array power function left>
  | <md-array power function right>
  | <md-array power function both>
```

```
<md-array power function left> ::=
  POWER <left paren> <md-array numeric expression> <comma>
      <numeric value expression> <right paren>

<md-array power function right> ::=
  POWER <left paren> <numeric value expression> <comma>
      <md-array numeric expression> <right paren>

<md-array power function both> ::=
  POWER <left paren> <md-array numeric expression> <comma>
      <md-array numeric expression> <right paren>

<md-array modulus expression> ::=
    <md-array modulus function left>
  | <md-array modulus function right>
  | <md-array modulus function both>

<md-array modulus function left> ::=
  MOD <left paren> <md-array numeric expression> <comma>
      <numeric value expression> <right paren>

<md-array modulus function right> ::=
  MOD <left paren> <numeric value expression> <comma>
      <md-array numeric expression> <right paren>

<md-array modulus function both> ::=
  MOD <left paren> <md-array numeric expression> <comma>
      <md-array numeric expression> <right paren>

<md-array absolute value expression> ::=
  ABS <left paren> <md-array numeric expression> <right paren>

<md-array natural logarithm> ::=
  LN <left paren> <md-array numeric expression> <right paren>

<md-array common logarithm> ::=
  LOG10 <left paren> <md-array numeric expression> <right paren>

<md-array exponential function> ::=
  EXP <left paren> <md-array numeric expression> <right paren>

<md-array square root> ::=
  SQRT <left paren> <md-array numeric expression> <right paren>

<md-array floor function> ::=
  FLOOR <left paren> <md-array numeric expression> <right paren>

<md-array ceiling function> ::=
  { CEIL | CEILING } <left paren> <md-array numeric expression> <right paren>

<md-array trigonometric function> ::=
  <trigonometric function name> <left paren> <md-array numeric expression> <right paren>

<md-array concatenation> ::=
  MDCONCAT <left paren> <md-array value expression> <comma> <md-array md-axis> <right paren>
```

Syntax Rules

1) If <md-array reshape function>, <md-array shift function>, or <md-array scale function> is specified, then:

   a) Let *AVE* be the <md-array value expression> immediately contained in <md-array reshape function>, <md-array shift function>, or <md-array scale function>. Let *MDT* be the declared

type of *AVE*, let *EDT* be the element type of *MDT*, let *d* be the MD-dimension of *MDT*, and let MN_i be the name of the *i*-th MD-axis in the maximum MD-extent of *MDT*, 1 (one) ≤ *i* ≤ *d*.

b) For all *i*, 1 (one) ≤ *i* ≤ *d*, let TN_i be MDAXIS_NAME(*AVE*, *i*), let TLO_i be MDAXIS_LOW(*AVE*, *i*), and let THI_i be MDAXIS_HIGH(*AVE*, *i*).

c) Case:

i) If <md-array shift function> is specified, then:

1) Case:

A) If the immediately contained <md-axis slice list> is an <md-axis slice list named>, then let *e* be the number of <md-axis slice named> simply contained in the <md-axis slice list named>. For *i*, 1 (one) ≤ *i* ≤ *e*, let SN_i be the <md-axis name> and let SP_i be the <numeric value expression> simply contained in the *i*-th <md-axis slice named>.

I) *e* shall be less than or equal to *d*.

II) For all *i* and *j*, *i* ≠ *j* and 1 (one) ≤ *i*, *j* ≤ *e*, SN_i shall not be equivalent to SN_j.

III) For all *i*, 1 (one) ≤ *i* ≤ *e* there shall exist *j*, 1 (one) ≤ *j* ≤ *d*, such that SN_i is equivalent to MN_j.

IV) For all *j*, 1 (one) ≤ *j* ≤ *d*

Case:

1) If there exists *i*, 1 (one) ≤ *i* ≤ *e*, such that SN_i is equivalent to MN_j, then let S_j be SP_i.

2) Otherwise, let S_j be 0 (zero).

B) Otherwise, let *e* be the number of <numeric value expression>s simply contained in the <md-axis slice list positional>. *e* shall be equal to *d*. For *i*, 1 (one) ≤ *i* ≤ *e*, let S_i be the <numeric value expression> simply contained in the *i*-th <md-axis slice positional>.

2) <md-array value function> shall not generally contain a <routine invocation> whose subject routine is an SQL-invoked routine that is possibly non-deterministic or that possibly modifies SQL-data.

3) <md-array value function> shall not generally contain a <table primary> that contains a <data change delta table>.

4) The <md-array value function> is equivalent to:

```
MDARRAY [ MN1 ( MDAXIS_LOW( AVE, 1) + S1 : MDAXIS_LOW( AVE, 1) + S1 ),
          ...,
          MNd ( MDAXIS_LOW( AVE, d) + Sd : MDAXIS_LOW( AVE, d) + Sd )
        ]
  ELEMENTS AVE[ MN1 - S1, ..., MNd - Sd ]
```

ii) Otherwise, let *ATL* be the immediately contained <md-axis limits list>.

1) Case:

A) If <md-axis limits list named> *ABLN* is immediately contained in *ATL*, then let *e* be the number of <md-axis limits named> immediately contained in *ABLN*. For 1 (one) ≤ *i* ≤ *e*, let SN_i be the <md-axis name>, let SLO_i be the <md-axis lower limit expression>, and let SHI_i be the <md-axis upper limit expression> simply contained in the *i*-th <md-axis limits named>.

I) For all *i* and *j*, *i* ≠ *j* and 1 (one) ≤ *i*, *j* ≤ *e*, SN_i shall not be equivalent to SN_j.

II) For all *j*, 1 (one) ≤ *j* ≤ *e*,

1) There shall exist *i*, 1 (one) ≤ *i* ≤ *d*, such that SN_j is equivalent to MN_i.

2) If SLO_j is not an <asterisk>, then let TLO_i be SLO_j.

3) If SHI_j is not an <asterisk>, then let THI_i be SHI_j.

B) If an <md-axis limits list positional> *ATLP* is immediately contained in *ATL*, then let *e* be the number of <md-axis limits positional> immediately contained in *ATLP*. *e* shall be equal to *d*.

I) For all *j*, 1 (one) ≤ *j* ≤ *e*,

1) Let ABP_i be the *i*-th <md-axis limits positional> immediately contained in *ATLP*.

2) Let SLO_i be the <md-axis lower limit expression> simply contained in ABP_i.

3) Let SHI_i be the <md-axis upper limit expression> simply contained in ABP_i.

II) For all *i*, 1 (one) ≤ *i* ≤ *d*,

1) If SLO_i is not an <asterisk>, then let TLO_i be SLO_i.

2) If SHI_i is not an <asterisk>, then let THI_i be SHI_i.

C) Otherwise:

I) Let *AE* be the <md-array extent> immediately contained in *ATL*, let *AVE1* be the <md-array value expression> immediately contained in *AE*, and let *e* be the MD-dimension of *AVE1*. *e* shall be equal to *d*.

II) Case:

1) If <md-array reshape function> is specified, then let *OP* be MDRESHAPE.

2) Otherwise, let *OP* be SCALE.

III) <md-array value function> is equivalent to

```
OP ( AVE,
     [ MDAXIS_NAME(AVE1, 1)
         ( MDAXIS_LOW(AVE1, 1) : MDAXIS_HIGH(AVE1, 1) ),
       ...,
       MDAXIS_NAME(AVE1, e)
         ( MDAXIS_LOW(AVE1, e) : MDAXIS_HIGH(AVE1, e) )
```

```
                    ]
                )
```

2) Case:

   A) If <md-array reshape function> is specified, then:

      I) <md-array value function> shall not generally contain a <routine invocation> whose subject routine is an SQL-invoked routine that is possibly non-deterministic or that possibly modifies SQL-data.

      II) <md-array value function> shall not generally contain a <table primary> that contains a <data change delta table>.

      III) The <md-array value function> is equivalent to:

```
MDARRAY [ TN1 ( TLO1 : THI1 ), ..., TNd ( TLOd : THId )
ELEMENTS
CASE WHEN MDAXIS_LOW(AVE, 1) <= TN1 AND
          TN1 <= MDAXIS_HIGH(AVE, 1) AND ... AND
          MDAXIS_LOW(AVE, d) <= TNd AND
          TNd <= MDAXIS_HIGH(AVE, d)
     THEN AVE[ TN1, ..., TNd ]
     ELSE NULL
     END
```

   B) Otherwise, for all i, 1 (one) $\leq i \leq d$, let R_i be (MDAXIS_HIGH (*AVE*, i) – MDAXIS_LOW (*AVE*, i) + 1) / ($THI_i - TLO_i + 1$).

      I) <md-array value function> shall not generally contain a <routine invocation> whose subject routine is an SQL-invoked routine that is possibly non-deterministic or that possibly modifies SQL-data.

      II) <md-array value function> shall not generally contain a <table primary> that contains a <data change delta table>.

      III) The <md-array value function> is equivalent to:

```
MDARRAY [ TN1 ( TLO1 : THI1 ), ...,
          TNd ( TLOd : THId ) ]
ELEMENTS AVE[ FLOOR((TN1 - TLO1) * R1) + SLO1, ...,
              FLOOR((TNd - TLOd) * Rd) + SLOd ]
```

2) If <md-array modulus expression> or <md-array power function> is specified, then let *AVF* be the <md-array modulus expression> or <md-array power function>.

   a) *AVF* shall not generally contain a <routine invocation> whose subject routine is an SQL-invoked routine that is possibly non-deterministic or that possibly modifies SQL-data nor a <table primary> that contains a <data change delta table>.

   b) Case:

      i) If *AVF* is an <md-array power function>, then let *E* be the <md-array power function left>, <md-array power function right>, or <md-array power function both> immediately contained in *AVF*. Let *OP* be POWER.

      ii) Otherwise, let *E* be the <md-array modulus function left>, <md-array modulus function right>, or <md-array modulus function both> immediately contained in *AVF*. Let *OP* be MOD.

c) Let *V1* be the first <md-array numeric expression> or <numeric value expression> immediately contained in *E*, and let *V2* be the second <md-array numeric expression> or <numeric value expression> immediately contained in *E*.

d) Let *T1* be the declared type of *V1*, and let *T2* be the declared type of *V2*.

e) Case:

i) If *T1* is an MD-array type, then let *MD1* be the maximum MD-extent of *T1*, let *d1* be the number of MD-axes in *MD1*, let $MN1_i$ be the name of the *i*-th MD-axis in *MD1*, 1 (one) ≤ *i* ≤ *d1*. Let *d* be *d1*, let *MD* be *MD1*, let MN_i be $MN1_i$, let *AVE* be *V1*, and let *EXP1* be *V1*[MN_1, ..., MN_d].

ii) Otherwise, let *EXP1* be *V1*.

f) Case:

i) If *T2* is an MD-array type, then let *MD2* be the maximum MD-extent of *T2*, let *d2* be the number of MD-axes in *MD2*, let $MN2_i$ be the name of the *i*-th MD-axis in *MD2*, 1 (one) ≤ *i* ≤ *d2*. Let *d* be *d2*, let *MD* be *MD2*, let MN_i be $MN2_i$, let *AVE* be *V2*, and let *EXP2* be *V2*[MN_1, ..., MN_d].

ii) Otherwise, let *EXP2* be *V2*.

g) If both *T1* and *T2* are MD-array types, then *d1* shall be equal to *d2* and $MN1_i$ shall be equivalent to $MN2_i$, 1 (one) ≤ *i* ≤ *d*.

h) The declared type of *E* is an MD-array type with maximum MD-extent *MD*.

i) If only one of *T1* or *T2* is an MD-array type, then *E* is equivalent to

```
MDARRAY [ MN1 ( MDAXIS_LOW (AVE, 1) : MDAXIS_HIGH (AVE, 1) ),
          ...,
          MNd ( MDAXIS_LOW (AVE, d) : MDAXIS_HIGH (AVE, d) ) ]
  ELEMENTS OP ( EXP1, EXP2 )
```

3) Let *E* be the <md-array absolute value expression>, <md-array natural logarithm>, <md-array common logarithm>, <md-array exponential function>, <md-array square root>, <md-array floor function>, <md-array ceiling function>, or <md-array trigonometric function> simply contained in <md-array value function>. Let *AVE* be the <md-array value expression> simply contained in *E*. Let *MDT* be the declared type of *AVE*, let *EDT* be the element type of *MDT*, let *d* be the MD-dimension of *MDT*, and let MN_i be the name of the *i*-th MD-axis in the maximum MD-extent of the declared type of *AVE*, 1 (one) ≤ *i* ≤ *d*.

a) Let *OP* be

Case:

i) If *E* is <md-array absolute value expression>, then ABS.

ii) If *E* is <md-array natural logarithm>, then LN.

iii) If *E* is <md-array common logarithm>, then LOG10.

iv) If *E* is <md-array exponential function>, then EXP.

v) If *E* is <md-array square root>, then SQRT.

vi) If *E* is <md-array floor function>, then FLOOR.

vii) If *E* is <md-array ceiling function>, then CEIL.

viii) If *E* is <md-array trigonometric function>, then the <trigonometric function name> immediately contained in *E*.

b) *E* shall not generally contain a <routine invocation> whose subject routine is an SQL-invoked routine that is possibly non-deterministic or that possibly modifies SQL-data.

c) *E* shall not generally contain a <table primary> that contains a <data change delta table>.

d) *E* is equivalent to

```
MDARRAY [ MN1 ( MDAXIS_LOW (AVE, 1) : MDAXIS_HIGH (AVE, 1) ), ...,
          MNd ( MDAXIS_LOW (AVE, d) : MDAXIS_HIGH (AVE, d) ) ]
ELEMENTS OP(AVE[ MN1, ..., MNd ])
```

4) If <md-array concatenation> *AC* is specified, then:

a) Let *AVE1* be the <md-array value expression> immediately contained in *AC*, let *d* be the MD-dimension of *AVE1*, and let MLO_i and MHI_i be the lower limit and upper limit, respectively, of the *i*-th MD-axis in the maximum MD-extent of the declared type of *AVE1*, 1 (one) ≤ *i* ≤ *d*. Let *AVE2* be the <md-array value expression> immediately contained in the <md-array md-axis> immediately contained in *AC*, let *e* be the MD-dimension of *AVE2*, and let ELO_i and EHI_i be the lower limit and upper limit, respectively, of the *i*-th MD-axis in the maximum MD-extent of the declared type of *AVE2*, 1 (one) ≤ *i* ≤ *d*. *d* shall be equal to *e*.

b) Let *DT1* be the element type *AVE1* and *DT2* be the element type of *AVE2*. The Syntax Rules of Subclause 9.5, "Result of data type combinations", are applied with the set comprising *DT1* and *DT2* as *DTSET*; let *DT* be the *RESTYPE* returned from the application of those Syntax Rules

c) Let *AD* be the MD-axis index specified by the <md-array md-axis> immediately contained in *AC*.

d) The declared type of the result of <md-array concatenation> is an MD-array type with element type *DT*, and a maximum MD-extent equal to the maximum MD-extent of the first input MD-array, except along the *AD*-th MD-axis, which has a lower limit equal to MLO_{AD}, and upper limit equal to MHI_{AD} + (EHI_{AD} - ELO_{AD} + 1).

Access Rules

No additional Access Rules.

General Rules

1) Case:

a) If <md-array concatenation> is specified, then let *AV1* be the value of the first <md-array value expression> and let *AV2* be the value of the second <md-array value expression>.

i) If at least one of *AV1* and *AV2* is the null value, then the result of the <md-array concatenation> is the null value.

ii) Let $N1_i$, $LO1_i$, and $HI1_i$, be the name, lower limit, and upper limit, respectively, of the *i*-th MD-axis in the MD-extent of *AV1*, 1 (one) ≤ *i* ≤ *d*. Let $N2_i$, $LO2_i$, and $HI2_i$, be the name, lower limit, and upper limit, respectively, of the *i*-th MD-axis in the MD-extent of *AV2*, 1 (one) ≤ *i* ≤ *d*.

iii) If $N1_{AD}$ is not equivalent to $N2_{AD}$, then an exception condition is raised: *data exception — MD-array invalid MD-axis (2203T)*.

iv) If there exists *i*, 1 (one) ≤ *i* ≤ *d* and *i* ≠ *AD*, such that $LO1_i \neq LO2_i$ or $HI1_i \neq HI2_i$, then an exception condition is raised: *data exception — MD-array operands with non-matching MD-extents (2203R)*.

v) For all *i*, 1 (one) ≤ *i* ≤ *d* and *i* ≠ *AD*, let $LO3_i$ be $LO1_i$ and let $HI3_i$ be $HI1_i$. Let $LO3_{AD}$ be $LO1_{AD}$ and $HI3_{AD}$ be $HI1_{AD}$ + ($HI2_{AD}$ - $LO2_{AD}$ + 1). Let *D3* be the MD-extent denoted by [$N1_1$ ($LO3_1$: $HI3_1$), ..., $N1_d$ ($LO3_d$: $HI3_d$)].

vi) The result value of <md-array concatenation> is equivalent to the value of:

```
COALESCE ( MDRESHAPE ( AV1, D3 ),
           MDRESHAPE ( MDSHIFT ( AV2, [ N1AD ( HI1AD + 1 ) ] ), D3 ) )
```

b) If <md-array modulus expression> or <md-array power function> is specified, then let *LD* be the MD-extent of the value of *AVE1* and let *RD* be the MD-extent of the value of *AVE2*.

Case:

i) If *LD* is not equal to *RD*, then an exception condition is raised: *data exception — MD-array operands with non-matching MD-extents (2203R)*.

ii) Otherwise, the value of *E* is equivalent to the value of:

```
MDARRAY MN1 ( MDAXIS_LOW(AVE, 1) : MDAXIS_HIGH(AVE, 1) ),
        ...
        MNd ( MDAXIS_LOW(AVE, d) : MDAXIS_HIGH(AVE, d) )
ELEMENTS OP ( EXP1, EXP2 )
```

Conformance Rules

1) Without Feature A010, "Multi-dimensional array support", conforming SQL language shall not contain <md-array value function>.

6.14 <md-array value constructor>

Function

Specify construction of an MD-array.

Format

```
<md-array value constructor> ::=
    <md-array value constructor by enumeration>
  | <md-array value constructor by query>
  | <md-array value constructor by iteration>
  | <md-array value constructor by decoding>
  | <md-array value constructor by join>

<md-array value constructor by enumeration> ::=
  MDARRAY <md-extent alternative> <md-array element list>

<md-array element list> ::=
  <left bracket or trigraph> <md-array element list inner> <right bracket or trigraph>

<md-array element list inner> ::=
  <md-array element> [ { <comma> <md-array element> }... ]

<md-array element> ::=
  <value expression>

<md-array value constructor by query> ::=
  MDARRAY <md-extent alternative> <table subquery>

<md-array value constructor by iteration> ::=
  MDARRAY <md-extent alternative>
      ELEMENTS <md-array element>

<md-array value constructor by decoding> ::=
  MDDECODE <left paren> <string value expression>
      <comma> <format identifier> <md-array returning clause> <right paren>

<md-array returning clause> ::=
  RETURNING <md-array type>

<md-array value constructor by join> ::=
  MDJOIN <left paren>
      <md-array value expression as field> <comma> <md-array value expression as field>
      [ { <comma> <md-array value expression as field> }... ] <right paren>

<md-array value expression as field> ::=
  <md-array value expression> [ AS <field name> ]
```


Syntax Rules

1) The declared type of <md-array value constructor> *AVC* is the declared type of the immediately contained <md-array value constructor by enumeration>, <md-array value constructor by query>, <md-array value constructor by iteration>, <md-array value constructor by decoding>, or <md-array value constructor by join>.

2) If <md-array value constructor by enumeration>, <md-array value constructor by query>, or <md-array value constructor by iteration> is specified, then:

a) Let *AVCE* be the <md-array value constructor by enumeration>, <md-array value constructor by query>, or <md-array value constructor by iteration>.

b) Let *EA* be the <md-extent alternative> immediately contained in *AVCE*.

c) Let *MD* be the maximum MD-extent specified by *EA*, let *d* be the number of MD-axes in *MD*, and let MN_i be the name of the *i*-th MD-axis in *MD*, 1 (one) ≤ *i* ≤ *d*.

3) If *AVC* is an <md-array value constructor by enumeration>, then the Syntax Rules of Subclause 9.5, "Result of data type combinations", are applied with the declared types of the <md-array element>s immediately contained in the <md-array element list inner> immediately contained in *AVCE* as *DTSET*; let *ET* be the *RESTYPE* returned from the application of those Syntax Rules. The declared type of *AVCE* is an MD-array type with element type *ET* and maximum MD-extent *MD*.

4) If *AVC* is an <md-array value constructor by query>, then:

a) The <query expression> *QE* simply contained in the <table subquery> shall be of degree *d* + 1 (one). For all *i*, 1 (one) ≤ *i* ≤ *d*:

i) There shall exist a unique *j*, 1 (one) ≤ *j* ≤ *N*, such that the name of the *j*-th column of *QE* is equivalent to MN_i.

ii) Let $NORD_i$ be the *j*-th column of *QE*. The declared type of $NORD_i$ shall be exact numeric with scale zero.

b) Let *NV* be the number, such that the *NV*-th column of *QE* is not equivalent to any $NORD_i$, 1 (one) ≤ *i* ≤ *d*.

c) Let *ET* be the declared type of the *NV*-th column of *QE*.

d) The declared type of *AVCE* is MD-array with element type *ET* and maximum MD-extent *MD*.

5) If *AVC* is an <md-array value constructor by iteration>, then:

a) Let *ET* be the declared type of the immediately contained <md-array element>.

b) The declared type of *AVCE* is an MD-array type with element type *ET* and maximum MD-extent *MD*.

c) For 1 (one) ≤ *i* ≤ *d*, let AV_i be an <md-axis variable> whose name is equivalent to MN_i. The scope of AV_i is the <md-array element> immediately contained in *AVC*.

6) If <md-array value constructor by decoding> *ACD* is specified, then:

a) Let *AT* be the <md-array type> simply contained in *ACD*. *AT* shall not contain <asterisk>.

b) Let *MD* be the maximum MD-extent of *AT* and let *ET* be the element type of *AT*.

c) Let *FID* be the value of the <format identifier> immediately contained in *ACD*. *FID* shall be a valid MIME identifier as specified in RFC 2046.

d) The declared type of *ACD* is *AT*.

7) If *AVC* is an <md-array value constructor by join>, then:

a) *AVC* shall not generally contain a <routine invocation> whose subject routine is an SQL-invoked routine that is possibly non-deterministic or that possibly modifies SQL-data.

b) *AVC* shall not generally contain a <table primary> that contains a <data change delta table>.

c) Let *N* be the number of <md-array value expression as field>s immediately contained in *AVC*. For all *i*, 1 (one) ≤ *i* ≤ *N*:

i) Let $AVCF_i$ be the i-th <md-array value expression as field> immediately contained in *AVC*.

ii) Either every $AVCF_i$ shall immediately contain a <field name>, or no $AVCF_i$ shall immediately contain a <field name>.

iii) Let AVE_i be the <md-array value expression> immediately contained in $AVCF_i$, let MDT_i be the declared type of AVE_i, and let MD_i be the maximum MD-extent of MDT_i.

iv) Let d be the MD-dimension of MD_1 and let MN_k be the name of the k-th MD-axis in MD_1, 1 (one) ≤ k ≤ d. The MD-dimension of MD_j, 2 ≤ j ≤ N, shall be equal to d, and the name of the k-th MD-axis in MD_j shall be equivalent to MN_k.

v) Let ET_i be the element type of MDT_i. ET_i shall not be a row type or structured type.

vi) Case:

1) If no $AVCF_i$ immediately contains a <field name>, then the Syntax Rules of Subclause 9.15, "Indexed name", are applied with *N* as *LASTINDEX*, *i* as *INDEX*, and 'FIELD' as *PREFIX*; let FN_i be the *INDEXEDNAME* returned from the application of those Syntax Rules

2) Otherwise, let FN_i be the <field name> immediately contained in $AVCF_i$.

d) Let *ERT* be a row type described by a sequence of (FN_i, ET_i) pairs, 1 (one) ≤ i ≤ N.

e) The declared type of *MAVC* is an MD-array type with element type *ERT* and maximum MD-extent MD_1.

Access Rules

None.

General Rules

1) The result of <md-array value constructor> *AVC* is the result of the simply contained <md-array value constructor by enumeration>, <md-array value constructor by query>, <md-array value constructor by iteration>, <md-array value constructor by decoding>, or <md-array value constructor by join>.

2) If <md-array value constructor by enumeration>, <md-array value constructor by query>, <md-array value constructor by iteration> is specified, then:

a) Let *AVCE* be the <md-array value constructor by enumeration>, <md-array value constructor by query>, <md-array value constructor by iteration>.

b) Let *EA* be the <md-extent alternative> immediately contained in *AVCE*.

c) Let *D* be the MD-extent specified by *EA* and let *d* be the number of MD-axes in *D*.

3) The result of <md-array value constructor by enumeration> is an MD-array *TV* with MD-extent *D*, determined as follows:

a) Let LO_i and HI_i be the lower and upper limits, respectively, of the i-th MD-axis in *D*, 1 (one) ≤ i ≤ d.

b) Let AE_i be the value of HI_i − LO_i + 1, 1 (one) ≤ i ≤ d.

c) Let *AEL* be the <md-array element list inner> simply contained in *AVCE*.

d) For each coordinate *R* within *D*:

i) Let P_i be the *i*-th element of *R*, 1 (one) ≤ *i* ≤ *d*.

ii) Let *j* be $(P_d - LO_d) + AE_d \times ((P_{d-1} - LO_{d-1}) + AE_{d-1} \times (\ldots + AE_2 \times (P_1 - LO_1)) \ldots)) + 1$.

iii) The element in *TV* at coordinate *R* is the value of the *j*-th <md-array element> immediately contained in *AEL*.

4) The result of <md-array value constructor by query> is an MD-array *TV* with MD-extent *D*, determined as follows:

a) *QE* is evaluated, producing a table *T*.

b) For all coordinates *W* within *D*, the element value in *TV* at coordinate *W* is initially set to the null value.

c) For each row in *T*:

i) Let P_i be the value of the $NORD_i$-th column, 1 (one) ≤ *i* ≤ *d*.

ii) If any P_i is the null value, then an exception condition is raised: *data exception — MD-array null coordinate in query constructor (2203N)*.

iii) Let *R* be the coordinate denoted by [P_1, ..., P_d]. If *R* is not within *D*, then an exception condition is raised: *data exception — MD-array coordinate not within specified MD-extent (2203P)*.

iv) A coordinate [P_1, ..., P_d] is equal to coordinate [Q_1, ..., Q_d] if $P_i = Q_i$, 1 (one) ≤ *i* ≤ *d*. If *R* is equal to the coordinate specified by any other row in *T*, then an exception condition is raised: *data exception — MD-array duplicate coordinate in query constructor (2203M)*.

v) The value in the *NV*-th column of *T* is the element value in *TV* at coordinate *R*.

5) The result of <md-array value constructor by iteration> is an MD-array *TV* with MD-extent *D*, determined as follows.

For each coordinate *R* within *D*:

a) Let P_i be the *i*-th element of *R*, 1 (one) ≤ *i* ≤ *d*.

b) The element in *TV* at coordinate *R* is the value of the <md-array element> immediately contained in the <md-array value constructor by iteration>, in which the value of any AV_i, 1 (one) ≤ *i* ≤ *d*, is equal to P_i.

6) If <md-array value constructor by decoding> *ACD* is specified, then let *SVE* be the <string value expression> immediately contained in *ACD*. Let *BP* be the result of JSON_QUERY(*SVE*, 'lax $data').

a) If *BP* is the null value, then the value of *ACD* is the null value and no further General Rules of this Subclause are applied.

b) Case:

i) If *FID* is equivalent to 'application/json', then:

1) Given an SQL/JSON item *J* and an MD-extent *D*, the function *F* is defined recursively as follows.

Case:

A) If *D* is the null value, then:

I) Let *ST1* be the completion condition *successful completion (00000)*.

II) The General Rules of Subclause 9.48, "Casting an SQL/JSON sequence to an SQL type", in ISO/IEC 9075-2, are applied with *ST1* as *STATUS IN*, *J* as *SQL/JSON SEQUENCE*, NULL as *EMPTY BEHAVIOR*, ERROR as *ERROR BEHAVIOR*, and *ET* as *DATA TYPE*; let *ST2* be the *STATUS OUT* and let *V* be the *VALUE* returned from the application of those General Rules.

III) If *ST2* is an exception condition, then the exception condition *ST2* is raised. Otherwise, the result of *F*(*J*, *D*) is *V*.

B) Otherwise:

I) Let *LO* and *HI* be the lower and upper limits of the first MD-axis of *D*, and let *EX* be *HI* − *LO* + 1.

II) If *J* is not an SQL/JSON array, or if the number of elements of *J* is not equal to *EX*, then an exception condition is raised: *data exception — MD-array decoding error (2203Y)*.

III) Case:

1) If the number of MD-axes in *D* is equal to 1 (one), then let *ND* be the null value.

2) Otherwise, let *ND* be the MD-extent obtained by removing the first MD-axis from *D*.

IV) Let J_k be the *k*-th element of *J* and let G_k be *F*(J_k, *ND*), 1 (one) ≤ *k* ≤ *EX*.

V) The result of *F*(*J*, *D*) is G_1, ..., G_k.

2) The General Rules of Subclause 9.42, "Parsing JSON text", in ISO/IEC 9075-2, are applied with *BP* as *JSON TEXT*, FORMAT JSON as *FORMAT OPTION*, and WITHOUT UNIQUE KEYS as *UNIQUENESS CONSTRAINT*; let *ST* be the *STATUS* and let *SJI* be the *SQL/JSON ITEM* returned from the application of those General Rules.

3) If *ST* is an exception condition, then the exception condition *ST* is raised.

4) Let *ELEMENTS* be the result of *F*(*SJI*, *MD*).

5) The result of *ACD* is:

```
MDARRAY MD [ ELEMENTS ]
```

ii) Otherwise, let *AVE* be an implementation-defined (IV098) MD-array value obtained by decoding *BP* according to the format indicated by *FID*. If a decoding error occurs, then an exception condition is raised: *data exception — MD-array decoding error (2203Y)*.

7) If *AVC* is an <md-array value constructor by join>, then:

a) Let AV_i be the value of AVE_i, 1 (one) ≤ *i* ≤ *N*, and let D_i be the MD-extent of AV_i. For all *j*, 2 ≤ *j* ≤ *N*, if D_j is not equal to D_1, then an exception condition is raised: *data exception — MD-array operands with non-matching MD-extents (2203R)*.

b) Let N_k, LO_k, and HI_k, 1 (one) ≤ *k* ≤ *d*, be the name, lower limit, and upper limit, respectively, of the *i*-th MD-axis in D_1.

c) The value of *AVC* is the value of:

```
MDARRAY [ N1 ( LO1 : HI1 ), ..., Nd ( LOd : HId ) ]
ELEMENTS ROW ( AVE1 [ N1, ..., Nk ], ..., AVEN [ N1, ..., Nk ] )
```

Conformance Rules

1) Without Feature A010, “Multi-dimensional array support”, conforming SQL language shall not contain an <md-array value constructor>.

2) Without Feature A013, “MD-array row element type support”, conforming SQL language shall not contain an <md-array value constructor> that immediately contains an <md-array value constructor by join>.

3) Without Feature A014, “MDA encode/decode functions media type: application/json”, conforming SQL language shall not contain <md-array value constructor by decoding>.

4) Without Feature A015, “MDA encode/decode functions with implementation-defined behavior”, conforming SQL language shall not contain an <md-array value constructor by decoding> that contains a <format identifier> that is not equivalent to 'application/json'.

6.15 <md-array element reference>

Function

Returns an element of an MD-array.

Format

```
<md-array element reference> ::=
  <md-array value expression> <left bracket or trigraph> <md-axis slice list> <right bracket
  or trigraph>

<md-axis slice list> ::=
    <md-axis slice list named>
  | <md-axis slice list positional>

<md-axis slice list named> ::=
  <md-axis slice named> [ { <comma> <md-axis slice named> }... ]

<md-axis slice list positional> ::=
  <md-axis slice positional> [ { <comma> <md-axis slice positional> }... ]
```

Syntax Rules

1) Let *AS* be the <md-array element reference>, let *AVE* be the <md-array value expression> immediately contained in *AS*, and let *ASL* be the <md-axis slice list> immediately contained in *AS*.

2) Let *MDT* be the declared type of *AVE* and let *EDT* be the element type of *MDT*.

3) The Syntax Rules of Subclause 9.17, "Canonicalize MD-array element reference", are applied with *MDT* as *MDARRAY TYPE* and *ASL* as *MDAXIS SLICE LIST*; let *CASL* be the *CANONICAL SLICE COORDINATE* returned from the application of those Syntax Rules.

4) The declared type of *AS* is *EDT*.

Access Rules

None.

General Rules

1) Let *AV* be the value of *AVE*.

2) If *AV* is the null value, then the result of *AS* is the null value and no further General Rules of this Subclause are applied.

3) Let *D* be the MD-extent of *AV*, let *d* be the number of MD-axes in *D*, and let LO_i and HI_i be the lower and upper limits, respectively, of the *i*-th MD-axis in *D*, 1 (one) ≤ *i* ≤ *d*.

4) Let P_i, 1 (one) ≤ *i* ≤ *d*, be the value of the *i*-th element of *CASL*. If P_i is less than LO_i or greater than HI_i, then an exception condition is raised: *data exception — MD-array element reference not within MD-extent (22040)*.

5) Let *SP* be the coordinate denoted by [P_1, ..., P_d].

6) The result of *AS* is the element in *AV* at coordinate *SP*.

Conformance Rules

1) Without Feature A010, "Multi-dimensional array support", conforming SQL language shall not contain an <md-array element reference>.

7 Query expressions

This Clause modifies Clause 7, "Query expressions", in ISO/IEC 9075-2.

7.1 <table reference>

This Subclause modifies Subclause 7.6, "<table reference>", in ISO/IEC 9075-2.

Function

Reference a table.

Format

```
<table primary> ::=
    !! All alternatives from ISO/IEC 9075-2
  | <md-array extent function> <correlation or recognition>

<md-array extent function> ::=
    <md-array extent>
  | <md-array max extent>

<md-array extent> ::=
  MDEXTENT <left paren> <md-array value expression> <right paren>

<md-array max extent> ::=
  MDMAX_EXTENT <left paren> <md-array value expression> <right paren>
```

Syntax Rules

1) Insert before SR 8): If *TP* simply contains <md-array extent function> *MEF*, then:

   a) *TP* is not generally updatable, simply updatable, effectively updatable, or insertable-into.

   b) A <correlation or recognition> immediately contained in *TP* shall immediately contain a <correlation name> *CN* and a <parenthesized derived column list> *DCLP*. *DCLP* shall contain exactly four <column name>s, none of which are equivalent to any of the others.

   c) Let *AVE* be the <md-array value expression> simply contained in *MEF*. Let *MDT* be the declared type of *AVE*, let *EDT* be the element type of *MDT*, and let *d* be the MD-dimension of *MDT*.

   d) If <md-array extent> is specified, then *MEF* is equivalent to:

```
LATERAL (
  SELECT *
  FROM UNNEST (
      ARRAY [ ROW ( 1, MDAXIS_NAME(AVE, 1),
                    MDAXIS_LOW(AVE, 1), MDAXIS_HIGH(AVE, 1) ),
        ...,
        ROW ( d, MDAXIS_NAME(AVE, d),
                    MDAXIS_LOW(AVE, d), MDAXIS_HIGH(AVE, d) ) ]
    ) AS CN DCLP
  WHERE AVE IS NOT NULL
)
```

e) If <md-array max extent> is specified, then let MN_i, MLO_i, and MHI_i, be the name, lower limit and upper limit, respectively, of the i-th MD-axis in the maximum MD-extent of *MDT*, 1 (one) ≤ i ≤ d. *MEF* is equivalent to:

```
( SELECT *
  FROM UNNEST ( ARRAY [ ROW (1, MN_1, MLO_1, MHI_1),
                        ...,
                        ROW (d, MN_d, MLO_d, MHI_d) ] )
       AS CN DCLP )
```

2) Augment SR 8) by adding "and no <collection value expression> immediately contained in *CDT* is an <md-array value expression>. " as an additional restriction to the predicate in the lead text of the Rule.

3) Insert before SR 9): If *TP* simply contains a <collection derived table> *CDT* and *CDT* immediately contains a <collection value expression> *CVE* that is an <md-array value expression>, then:

a) *CDT* shall not immediately contain any other <collection value expression>s.

b) Let *MDT* be the declared type of *CVE*, and let *ET* be the element type of *MDT*.

c) Let *MD* be the maximum MD-extent of *MDT*, let *d* be the number of MD-axes in *MD*, and let MN_i be the name of the i-th MD-axis in *MD*, 1 (one) ≤ i ≤ d.

d) Case:

i) If ET_1 is a row type, then let *DET* be the degree of ET_1.

ii) Otherwise, let *DET* be 1 (one).

e) Case:

i) If WITH ORDINALITY is specified, then let *NORD* be 1 (one).

ii) Otherwise, let *NORD* be 0 (zero).

f) Let *TPTC* be the table specified by *CDT*.

g) The degree of *TPTC* is *NORD* + *d* + *DET*.

h) Let *ORD* and *EN* be implementation-dependent (UV005) <column name>s that are not equivalent to MN_j, 1 (one) ≤ j ≤ d, or any <identifier> contained in *TP*.

i) The column descriptors of each of the columns of *TPTC* are determined as follows:

i) If WITH ORDINALITY is specified, then the name of the first column is *ORD* and the declared type is exact numeric with scale 0 (zero).

ii) The name of the j-th column, *NORD* + 1 (one) ≤ j ≤ *NORD* + *d*, is MN_{j-NORD} and the declared type is exact numeric with scale 0 (zero).

iii) Let *NEN* be *NORD* + *d*.

Case:

1) If *ET* is a row type, then the name and declared type of the k-th column, *NEN* + 1 ≤ k ≤ *NEN* + *DET* is the name and declared type, respectively, of the (k − *NEN*)-th field of *ET*.

2) Otherwise, the name of the (*NEN* + 1)-th column is *EN*, and the declared type is *ET*.

4) Augment SR 12) by adding "<collection derived table> that immediately contains a <collection value expression> that is an <md-array value expression> " to the list of BNF non-terminals that may specify TPTI in the lead text of the Rule.

5) Augment SR 12)b)i)2) by adding "<collection derived table> that immediately contains a <collection value expression> that is an <md-array value expression> " to the list of BNF non-terminals that may specify the table in the lead text of the rule.

Access Rules

No additional Access Rules.

General Rules

1) Insert before GR 7): If a <table primary> *TP* immediately contains <collection derived table> *CDT* that immediately contains a single <collection value expression> *CVE* that is an MD-array, then let *AVE* be the value of *CVE*. Let *D* be the MD-extent of *AVE*. The result of *TP* is a table value of cardinality equal to the cardinality of *D*, the rows of which are set as follows:

   a) Let *d* be the number of MD-axes in *D*, let LO_i and HI_i be the lower and upper limits, respectively, of the *i*-th MD-axis in *D*, and let AE_i be the value of $HI_i - LO_i + 1$, 1 (one) $\leq i \leq d$.

   b) For each coordinate *R* within *D*:

      i) Let *E* be the element of *AVE* at coordinate *R*.

      ii) Let P_i be the *i*-th element of *R*.

      iii) Let *IND* be the value of

$$\begin{aligned}&(\ P_d - LO_d \ ) + AE_d \\ &\times (\ (\ P_{d-1} - LO_{d-1} \ ) + AE_{d-1} \\ &\times (\ \ldots + AE_2 \\ &\times (\ P_1 - LO_1 \ ) \ ) \ldots \ ) + 1\end{aligned}$$

      iv) The values in each column of the *IND*-th row are determined as follows:

         1) If WITH ORDINALITY is specified, then the value in the first column is *IND*.

         2) The value in the *j*-th column, $NORD + 1 \leq j \leq NORD + d$, is P_i.

         3) Case:

            A) If *ET* is a row type, then the value of the *k*-th column, $NEN + 1 \leq k \leq NEN + DET$ is the $(k - NEN)$-th field of *E*.

            B) Otherwise, the value of the $(NEN + 1)$-th column is *E*.

Conformance Rules

1) Insert into CR 2), after the last list item:

   a) Feature A010, "Multi-dimensional array support".

2) Insert after the last CR: Without Feature A010, "Multi-dimensional array support", conforming SQL language shall not contain <md-array extent> or <md-array max extent>.

7.2 <query specification>

This Subclause modifies Subclause 7.16, "<query specification>", in ISO/IEC 9075-2.

Function

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

Format

No additional Format items.

Syntax Rules

1) Insert after SR 19)b)xiv): An <md-array element reference>.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

8 Predicates

This Clause modifies Clause 8, "Predicates", in ISO/IEC 9075-2.

8.1 <distinct predicate>

This Subclause modifies Subclause 8.15, "<distinct predicate>", in ISO/IEC 9075-2.

Function

Specify a test of whether two row values are distinct.

Format

No additional Format items.

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

1) Insert after GR 1)c)v): If *V1* and *V2* are values of an MD-array type, then

Case:

a) If *V1* and *V2* do not have interval-equal MD-extents, then the result is *True*.

b) If *V1* and *V2* have interval-equal MD-extents and there exists at least one coordinate *P* such that the element at coordinate *P* in *V1* is distinct from the element at coordinate *P* in *V2*, then the result is *True*.

c) Otherwise, the result is *False*.

Conformance Rules

No additional Conformance Rules.

9 Additional common rules

This Clause modifies Clause 9, "Additional common rules", in ISO/IEC 9075-2.

9.1 Retrieval assignment

This Subclause modifies Subclause 9.1, "Retrieval assignment", in ISO/IEC 9075-2.

Function

Specify rules for assignments to targets that do not support null values or that support null values with indicator parameters (e.g., assigning SQL-data to host parameters or host variables).

Subclause Signature

```
"Retrieval assignment" [General Rules] (
  Parameter: "TARGET",
  Parameter: "VALUE"
)
```

TARGET — a site that is the target of a retrieval assignment operation.

VALUE — a value that is to be assigned to TARGET.

Syntax Rules

1) Insert after SR 8)b): If *TD* is an MD-array type, then *SD* shall be an MD-array type. The MD-dimension of *TD* shall be equal to the MD-dimension of *SD*.

Access Rules

None.

General Rules

1) Replicate GR 1): Let *T* be the *TARGET* and let *V* be the *VALUE* in an application of the General Rules of this Subclause.

2) Insert after GR 7)v): If the declared type of *T* is an MD-array type, then

Case:

a) If the MD-extent *M* of *V* is strictly within the maximum MD-extent of *T*, then, for each pair of elements of *V* and *T*, the General Rules of this Subclause are applied with the element of *T* as *TARGET* and the element of *V* as *VALUE*. The MD-extent of the value of *T* is *M*.

b) Otherwise, an exception condition is raised: *data exception — MD-array source MD-extent not strictly within maximum target MD-extent (2203Q)*.

Conformance Rules

None.

9.2 Store assignment

This Subclause modifies Subclause 9.2, "Store assignment", in ISO/IEC 9075-2.

Function

Specify rules for assignments where the target permits null without the use of indicator parameters or indicator variables, such as storing SQL-data or setting the value of SQL parameters.

Subclause Signature

```
"Store assignment" [General Rules] (
  Parameter: "TARGET",
  Parameter: "VALUE"
)
```

TARGET — a site that is the target of a retrieval assignment operation.

VALUE — a value that is to be assigned to TARGET.

Syntax Rules

1) Insert after SR 6)b): If *TD* is an MD-array type, then *SD* shall be an MD-array type. The MD-dimension of *TD* shall be equal to the MD-dimension of *SD*.

Access Rules

No additional Access Rules.

General Rules

1) Replicate GR 1): Let *T* be the *TARGET* and let *V* be the *VALUE* in an application of the General Rules of this Subclause.

2) Insert after GR 3)b)xxii): If the declared type of *T* is an MD-array type, then

Case:

a) If the MD-extent *M* of *V* is strictly within the maximum MD-extent of *T*, then, for each pair of elements of *V* and *T*, the General Rules of this Subclause are applied with the element of *T* as *TARGET* and the element of *V* as *VALUE*. The MD-extent of the value of *T* is *M*.

NOTE 2 — The maximum MD-extent *L* of *T* is unchanged.

b) Otherwise, an exception condition is raised: *data exception — MD-array source MD-extent not strictly within maximum target MD-extent (2203Q)*.

Conformance Rules

No additional Conformance Rules.

9.3 Passing a value from a host language to the SQL-server

This Subclause modifies Subclause 9.3, "Passing a value from a host language to the SQL-server", in ISO/IEC 9075-2.

Function

Specify rules to pass a value from a host language to the SQL-server.

Subclause Signature

```
"Passing a value from a host language to the SQL-server" [General Rules] (
  Parameter: "LANGUAGE",
  Parameter: "SQL TYPE",
  Parameter: "HOST VALUE"
) Returns: "SQL VALUE"
```

LANGUAGE — the name of a host programming language (ADA, C, COBOL, FORTRAN, M, PASCAL, PLI).

SQL TYPE — a specification of an SQL data type, possibly including a <locator indication>.

HOST VALUE — a value of a host language type.

SQL VALUE — the SQL TYPE representation of HOST VALUE, that is the result of invoking this subclause using this signature.

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

1) Replicate GR 1): Let *LANG* be the *LANGUAGE*, let *DT* be the *SQL TYPE*, and let *PI* be the *HOST VALUE* in an application of the General Rules of this Subclause. The result of the application of this Subclause is returned as *SQL VALUE*.

2) Augment GR 5)a)ii) by adding "MD-array value" to the list of possible values of SV.

Conformance Rules

No additional Conformance Rules.

9.4 Passing a value from the SQL-server to a host language

This Subclause modifies Subclause 9.4, "Passing a value from the SQL-server to a host language", in ISO/IEC 9075-2.

Function

Specify rules to pass a value from the SQL-server to a host language.

Subclause Signature

```
"Passing a value from the SQL-server to a host language" [General Rules] (
  Parameter: "LANGUAGE",
  Parameter: "SQL TYPE",
  Parameter: "SQL VALUE"
) Returns: "HOST VALUE"
```

LANGUAGE — the name of a host programming language (ADA, C, COBOL, FORTRAN, M, PASCAL, PLI).

SQL TYPE — a specification of an SQL data type, possibly including a <locator indication>.

SQL VALUE — a value of type SQL TYPE.

HOST VALUE — a host language data type representation of SQL VALUE that is the result of invoking this subclause using this signature.

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

1) Replicate GR 1): Let *LANG* be the *LANGUAGE*, let *DT* be the *SQL TYPE*, and let *SV* be the *SQL VALUE* in an application of the General Rules of this Subclause. The result of the application of this Subclause is returned as *HOST VALUE*.

2) Augment GR 5)a) by adding "MD-array locator value" to the list of possible values of PI.

Conformance Rules

No additional Conformance Rules.

9.5 Result of data type combinations

This Subclause modifies Subclause 9.5, "Result of data type combinations", in ISO/IEC 9075-2.

Function

Specify the result data type of the result of certain combinations of values of compatible data types, such as <case expression>s, <collection value expression>s, or a column in the result of a <query expression>.

Subclause Signature

```
"Result of data type combinations" [Syntax Rules] (
  Parameter: "DTSET"
) Returns: "RESTYPE"
```

DTSET — a set of SQL data types.

RESTYPE — an SQL data type that can be used to represent values of all SQL data types contained in DTSET.

Syntax Rules

1) Replicate GR 1): Let *IDTS* be the *DTSET* in an application of the Syntax Rules of this Subclause. The result of the application of this Subclause is returned as *RESTYPE*.

2) Insert after SR 3)i): If any data type in *DTS* is an MD-array type, then every data type in *DTS* shall be an MD-array type of the same MD-dimension *d* and for all *i*, 1 (one) ≤ *i* ≤ *d*, the name of the *i*-th MD-axis of the MD-extent of some data type in *DTS* shall be equivalent to the name of the *i*-th MD-axis of the MD-extents of all other data types in *DTS*. The data type of the result is an MD-array type with element data type *ETR*, where *ETR* is the data type resulting from the application of this Subclause to the set of element types of the MD-array types of *DTS*, and maximum MD-extent equal to the union of the maximum MD-extents of the data types in *DTS*.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

9.6 Type name determination

This Subclause modifies Subclause 9.9, "Type name determination", in ISO/IEC 9075-2.

Function

Determine an <identifier> given the name of a predefined or collection type.

Syntax Rules

1) Insert after SR 2)w): If *DT* specifies MDARRAY, then let *FNSDT* be "MDARRAY".

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

9.7 Determination of identical values

This Subclause modifies Subclause 9.10, "Determination of identical values", in ISO/IEC 9075-2.

Function

Determine whether two instances of values are identical, that is to say, are occurrences of the same value.

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

1) Insert after GR 2)d)iii): If *V1* and *V2* are MD-arrays and have equal MD-extents and elements in the same coordinate in the two MD-arrays are identical, then *V1* is identical to *V2*.

Conformance Rules

No additional Conformance Rules.

9.8 Equality operations

This Subclause modifies Subclause 9.11, "Equality operations", in ISO/IEC 9075-2.

Function

Specify the prohibitions and restrictions by data type on operations that involve testing for equality.

Syntax Rules

1) Insert before SR 3): The declared type of an operand of an equality operation shall not be MD-array-ordered.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

9.9 Grouping operations

This Subclause modifies Subclause 9.12, "Grouping operations", in ISO/IEC 9075-2.

Function

Specify the prohibitions and restrictions by data type on operations that involve grouping of data.

Syntax Rules

1) Augment SR 3) by adding "MD-array-ordered" to the list of excluded declared types.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

9.10 Multiset element grouping operations

This Subclause modifies Subclause 9.13, "Multiset element grouping operations", in ISO/IEC 9075-2.

Function

Specify the prohibitions and restrictions by data type on the declared element type of a multiset for operations that involve grouping the elements of a multiset.

Syntax Rules

1) Augment SR 3) by adding "MD-array-ordered" to the list of excluded declared types.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

9.11 Ordering operations

This Subclause modifies Subclause 9.14, "Ordering operations", in ISO/IEC 9075-2.

Function

Specify the prohibitions and restrictions by data type on operations that involve ordering of data.

Syntax Rules

1) Augment SR 3) by adding "MD-array-ordered" to the list of excluded declared types.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

9.12 Potential sources of non-determinism

This Subclause modifies Subclause 9.16, "Potential sources of non-determinism", in ISO/IEC 9075-2.

Function

Define the potential sources of non-determinism.

Syntax Rules

1) Insert after SR 1)as): An <md-array value constructor by query>.

Access Rules

None.

General Rules

None.

Conformance Rules

None.

9.13 Invoking an SQL-invoked routine

This Subclause modifies Subclause 9.18, "Invoking an SQL-invoked routine", in ISO/IEC 9075-2.

Function

Execute the invocation of an SQL-invoked routine.

Format

No additional Format items.

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

NOTE 3 — The General Rules of this Subclause as defined in GR 2) of Subclause 9.18, "Invoking an SQL-invoked routine", in ISO/IEC 9075-2 are not always terminated when an exception condition is raised.

1) Insert before GR 9)g)ii)5): If *R* is an MD-array-returning external function, then:

   a) Let *MDAR* be an MD-array whose declared type is the result data type of *R*.

   b) The General Rules of Subclause 9.18, "Execution of MD-array-returning external functions", are applied with *MDAR* as *MDARRAY*, *ESPL* as *EFFECTIVE SQL PARAMETER LIST*, and *P* as *PROGRAM*.

2) Insert after GR 9)i)i)4)A)III): If *R* is an MD-array-returning external function, and *R* specifies PARAMETER STYLE SQL, then let *ERDI* be *MDAR*.

Conformance Rules

No additional Conformance Rules.

9.14 Data type identity

This Subclause modifies Subclause 9.30, "Data type identity", in ISO/IEC 9075-2.

Function

Determine whether two data types are compatible and have the same characteristics.

Syntax Rules

1) Insert after SR 11)b): If *PM* is an MD-array type, then the maximum MD-extent of *PM* shall be equal to the maximum MD-extent of *P*.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

9.15 Indexed name

Function

Specify rules to derive a name based on the given prefix and index.

Subclause Signature

```
"Indexed name" [Syntax Rules] (
  Parameter: "LASTINDEX",
  Parameter: "INDEX",
  Parameter: "PREFIX"
) Returns: "INDEXEDNAME"
```

LASTINDEX — an integer value specifying the largest index value for a given MD-array axis; it is used in this Subclause to determine a padding for the INDEX, so that MD-axes of the same MD-extent are assigned INDEXEDNAME values with a consistent length.

INDEX — an unsigned integer value specifying the (1-based) index of an anonymous MD-array axis within a particular MD-extent for which an indexed name is to be derived.

PREFIX — a character string value used to ensure that INDEXEDNAME is a valid SQL <identifier>.

INDEXEDNAME — the derived unique name (<identifier>) of an MD-array axis.

Syntax Rules

1) Let *D* be the *LASTINDEX*, let *AI* be the *INDEX*, and let *P* be the *PREFIX* in an application of the Syntax Rules of this Subclause. The result of the application of this Subclause is returned as *INDEXEDNAME*.

2) Let *NZ* be the result of

   ```
   FLOOR( LOG10( D ) ) - FLOOR( LOG10( AI ) )
   ```

3) Let PAD_0 be the zero-length string. Let PAD_i, 1 (one) ≤ i ≤ *NZ*, be the value of

   PAD_{i-1} || '0'

4) Let *AN* be the value of

   P || PAD_{NZ} || CAST(*AI* AS CHARACTER VARYING(*n*))

   where *n* is the implementation-defined (IL006) maximum length of a variable-length character string.

5) Evaluation of the Syntax Rules is terminated and control is returned to the invoking Subclause, which receives *AN* as *INDEXEDNAME*.

Access Rules

None.

General Rules

None.

Conformance Rules

None.

9.16 MD-array subset

Function

Convert an <md-array subset> to MD-extents that take into account sliced MD-axes.

Subclause Signature

```
"MD-array subset" [Syntax Rules] (
  Parameter: "MDARRAY TYPE",
  Parameter: "MDAXIS SUBSET LIST"
) Returns: "TARGET MAXIMUM MDEXTENT" and "EQUIVALENCE"
```

MDARRAY TYPE — an MD-array type.

MDAXIS SUBSET LIST — an <md-axis subset list>.

TARGET MAXIMUM MDEXTENT — a potentially empty maximum MD-extent derived from the provided MDARRAY TYPE and MDAXIS SUBSET LIST.

EQUIVALENCE — a potentially empty character string representing a syntax equivalence to a maximum MD-extent derived from the provided MDARRAY TYPE and MDAXIS SUBSET LIST.

```
"MD-array subset" [General Rules] (
  Parameter: "MDARRAY VALUE",
  Parameter: "MDAXIS SUBSET LIST"
) Returns: "SOURCE MDEXTENT", "TARGET MDEXTENT", and "SLICED MDAXES"
```

MDARRAY VALUE — an MD-array value.

MDAXIS SUBSET LIST — an <md-axis subset list>.

SOURCE MDEXTENT — the MD-extent of MDARRAY VALUE with the MDAXIS SUBSET LIST applied to it such that a pair of <md-axis lower limit expression> and <md-axis upper limit expression> replace the lower and upper limit of the corresponding MD-axis in the result unless equivalent to an <asterisk>, and a <md-axis slice positional> replaces both the lower and upper limit of the corresponding MD-axis.

TARGET MDEXTENT — same as the SOURCE MDEXTENT, except that it does not contain the MD-axes corresponding to <md-axis slice positional>s in the provided MDAXIS SUBSET LIST.

SLICED MDAXES — a potentially empty set of indexes of MD-axes that are slices in the provided MDAXIS SUBSET LIST.

Syntax Rules

1) Let *MDT* be the *MDARRAY TYPE* and let *ASL* be the *MDAXIS SUBSET LIST* in an application of the Syntax Rules of this Subclause. The result of the application of this Subclause is returned as *TARGET MAXIMUM MDEXTENT* and *EQUIVALENCE*.

2) *MDT* shall be an MD-array type and *ASL* shall be an <md-axis subset list>.

3) Let *EDT* be the element type of *MDT*, let *MD* be the maximum MD-extent of *MDT*, let *d* be the number of MD-axes in *MD*, and let MN_i, MLO_i, and MHI_i, be the name, lower limit, and upper limit, respectively, of the *i*-th MD-axis in *MD*, 1 (one) ≤ *i* ≤ *d*.

4) The declared type of <numeric value expression>s simply contained in *ASL* shall be exact numeric with scale 0 (zero).

5) Case:

a) If *ASL* is an <md-axis subset list positional>, then:

i) The number of <md-axis subset positional>s immediately contained in the <md-axis subset list positional> *ASLP* shall be *d*.

ii) *ASLP* shall simply contain at least one <md-axis limits positional>.

iii) Let ASP_i be the *i*-th <md-axis subset positional> immediately contained in *ASLP*, 1 (one) ≤ *i* ≤ *d*.

iv) Let *TMD* be a zero-length character string, and let *EQ* be a character string equivalent to

```
[ MN1 ( ASP1 ),
  ...,
  MNd ( ASPd )
]
```

b) If *ASL* is an <md-array subset extent>, then:

i) Let *AVE1* be the <md-array value expression> immediately contained in *ASL* and let *e* be the MD-dimension of *AVE1*. *e* shall be equal to *d*.

ii) Let *TMD* be a zero-length character string, and let *EQ* be a character string equivalent to

```
[ MDAXIS_LOW(AVE1, 1) : MDAXIS_HIGH(AVE1, 1),
  ...,
  MDAXIS_LOW(AVE1, e) : MDAXIS_HIGH(AVE1, e)
]
```

c) Otherwise:

i) Let *ASNNUM* be the number of <md-axis subset named> immediately contained in *ASL*. *ASNNUM* shall be less than or equal to *d*.

ii) For *i*, 1 (one) ≤ *i* ≤ *ASNNUM*:

1) Let ASN_i be the *i*-th <md-axis subset named> immediately contained in *ASL*.

2) Let M_i be the <md-axis name> simply contained in ASN_i.

3) Let O_i be the <md-interval expression> or <md-axis slice positional> simply contained in ASN_i.

iii) For all *i* and *j*, 1 (one) ≤ *i*, *j* ≤ *ASNNUM*, and *i* ≠ *j*, M_i shall not be equivalent to M_j.

iv) For all *i*, 1 (one) ≤ *i* ≤ *ASNNUM*, there shall exist *j*, 1 (one) ≤ *j* ≤ *d*, such that M_i is equivalent to MN_j.

v) Let *SLICENUM* be the number of <md-axis subset named> immediately contained in *ASL* that are an <md-axis slice named>.

1) *d* shall not be equal to *SLICENUM*.

2) Let *t* be *d* − *SLICENUM*.

3) Let *i* be 1 (one).

4) For all *j*, 1 (one) ≤ *j* ≤ *d*, such that no *k* exists, 1 (one) ≤ *k* ≤ *ASNNUM*, where MN_j is equivalent to M_k and O_k is an <md-axis slice positional>:

A) Let TMN_i be MN_j, let $TMLO_i$ be MLO_j, and let $TMHI_i$ be MHI_j.

B) *i* is increased by 1 (one).

vi) Let *TMD* be the maximum MD-extent denoted by [TMN_1 ($TMLO_1$: $TMHI_1$), ..., TMN_t ($TMLO_t$: $TMHI_t$)], and let *EQ* be a zero-length character string.

6) Evaluation of the Syntax Rules is terminated and control is returned to the invoking Subclause, which receives *TMD* as *TARGET MAXIMUM MDEXTENT* and *EQ* as *EQUIVALENCE*.

Access Rules

None.

General Rules

1) Let *AV* be the *MDARRAY VALUE* and let *ASL* be the *MDAXIS SUBSET LIST* in an application of the General Rules of this Subclause. The result of the application of this Subclause is returned as *SOURCE MDEXTENT*, *TARGET MDEXTENT*, and *SLICED MDAXES*.

2) Let *D* be the MD-extent of *AV*, let *d* be the number of MD-axes in *D*, and let MN_i, LO_i, and HI_i, be the name, lower limit, and upper limit, respectively, of the *i*-th MD-axis in *D*, 1 (one) ≤ *i* ≤ *d*.

3) Let *SLICEDAXES* be an empty set.

4) Let *ASNNUM* be the number of <md-axis subset named> simply contained in *ASL*.

NOTE 4 — *ASL* is an <md-axis subset list> that is an <md-axis subset list named>. The Syntax Rules of this Subclause syntactically transform <md-axis subset list positional> and <md-array subset extent> into a form that is an <md-axis subset list named>.

5) For *i*, 1 (one) ≤ *i* ≤ *ASNNUM*:

a) Let ASN_i be the *i*-th <md-axis subset named> simply contained in *ASL*.

b) Let M_i be the <md-axis name> simply contained in ASN_i.

c) Let O_i be the <md-interval expression> or <md-axis slice positional> simply contained in ASN_i.

6) Let *SLICENUM* be the number of <md-axis subset named> simply contained in *ASL* that are an <md-axis slice named>.

7) Let *t* be *d* – *SLICENUM*.

8) Let *i* be 1 (one).

9) For all *j*, 1 (one) ≤ *j* ≤ *d*:

Case:

a) If *k* exists, 1 (one) ≤ *k* ≤ *ASNNUM*, such that MN_j is equivalent to M_k, then

Case:

i) If O_k is an <md-interval expression>, then:

1) Let *PLO* be the <md-axis lower limit expression> immediately contained in O_k, and let *PHI* be the <md-axis upper limit expression> immediately contained in O_k.

2) Case:

A) If *PLO* is <asterisk>, then let SLO_j be LO_j.

B) Otherwise, let SLO_j be the value of *PLO*.

3) Case:

A) If *PHI* is <asterisk>, then let SHI_j be HI_j.

B) Otherwise, let SHI_j be the value of *PHI*.

4) If SLO_j or SHI_j is the null value, then an exception condition is raised: *data exception — MD-array null limit in subset (2203J)*.

5) Let TN_i be MN_j, let TLO_i be SLO_j, and let THI_i be SHI_j.

6) i is increased by 1 (one).

ii) Otherwise:

1) Let *P* be the value of the <md-axis slice positional> O_k.

2) If *P* is the null value, then an exception condition is raised: *data exception — MD-array null limit in subset (2203J)*.

3) Let SLO_j be *P* and let SHI_j be *P*.

4) j is added to *SLICEDAXES*.

b) Otherwise:

i) Let SLO_j be LO_j, let SHI_j be HI_j, let TN_i be MN_j, let TLO_i be SLO_j, and let THI_i be SHI_j.

ii) i is increased by 1 (one).

10) Let *SD* be the MD-extent denoted by [MN_1 (SLO_1 : SHI_1), ..., MN_d (SLO_d : SHI_d)], and let *TD* be the MD-extent denoted by [TN_1 (TLO_1 : THI_1), ..., TN_t (TLO_t : THI_t)].

11) Evaluation of the General Rules is terminated and control is returned to the invoking Subclause, which receives *SD* as *SOURCE MDEXTENT*, *TD* as *TARGET MDEXTENT*, and *SLICEDAXES* as *SLICED MDAXES*.

Conformance Rules

None.

9.17 Canonicalize MD-array element reference

Function

Convert an MD-array element reference to a canonical coordinate.

Subclause Signature

```
"Canonicalize MD-array element reference" [Syntax Rules] (
  Parameter: "MDARRAY TYPE",
  Parameter: "MDAXIS SLICE LIST"
) Returns: "CANONICAL SLICE COORDINATE"
```

MDARRAY TYPE — an MD-array type.

MDAXIS SLICE LIST — an <md-axis slice list>.

CANONICAL SLICE COORDINATE — a coordinate derived from the provided MDAXIS SLICE LIST such that all <md-axis slice positional> from it have been ordered to match the corresponding MD-axes in the provided MDARRAY TYPE.

Syntax Rules

1) Let *MDT* be the *MDARRAY TYPE* and let *ASL* be the *MDAXIS SLICE LIST* in an application of the Syntax Rules of this Subclause. The result of the application of this Subclause is returned as *CANONICAL SLICE COORDINATE*.

2) *MDT* shall be an MD-array type and *ASL* shall be an <md-axis slice list>.

3) Let *EDT* be the element type of *MDT*, let *MD* be the maximum MD-extent of *MDT*, let *d* be the number of MD-axes in *MD*, and let MN_i be the name of the *i*-th MD-axis in *MD*, 1 (one) ≤ *i* ≤ *d*.

4) Let *e* be the number of <md-axis slice named>s or <md-axis slice positional>s simply contained in *ASL*. *e* shall be equal to *d*.

   Case:

   a) If *ASL* is an <md-axis slice list positional> *ASLP*, then let O_i be the *i*-th <md-axis slice positional> immediately contained in *ASLP*, 1 (one) ≤ *i* ≤ *e*.

   b) Otherwise, *ASL* is an <md-axis slice list named> *ASLN*.

      i) Let ASN_i be the *i*-th <md-axis slice named> immediately contained in *ASLN* and let M_i be the <md-axis name> immediately contained in ASN_i, 1 (one) ≤ *i* ≤ *e*.

      ii) For all *i* and *j*, 1 (one) ≤ *i*, *j* ≤ *e*, and *i* ≠ *j*, M_i shall not be equivalent to M_j.

      iii) For all *i*, 1 (one) ≤ *i* ≤ *e*:

         1) There shall exist exactly one *j*, 1 (one) ≤ *j* ≤ *d*, such that M_i is equivalent to MN_j.

         2) Let O_i be the <md-axis slice positional> immediately contained in ASN_j.

5) Let *RET* be the coordinate denoted by [O_1, ..., O_d].

6) Evaluation of the Syntax Rules is terminated and control is returned to the invoking Subclause, which receives *RET* as *CANONICAL SLICE COORDINATE*.

Access Rules

None.

General Rules

None.

Conformance Rules

None.

9.18 Execution of MD-array-returning external functions

Function

Specify the execution of an external function that returns an MD-array value.

Subclause Signature

```
"Execution of MD-array-returning external functions" [General Rules] (
  Parameter: "MDARRAY",
  Parameter: "EFFECTIVE SQL PARAMETER LIST",
  Parameter: "PROGRAM"
)
```

MDARRAY — a coordinate derived from the provided MDAXIS SLICE LIST such that all <md-axis slice positional> from it have been ordered to match the corresponding MD-axes in the provided MDARRAY TYPE.

EFFECTIVE SQL PARAMETER LIST — effective SQL parameter list of the MD-array-returning external function.

PROGRAM — the program identified by the external name of the MD-array-returning external function.

Syntax Rules

None.

Access Rules

None.

General Rules

1) Let *AR* be the *MDARRAY*, let *ESPL* be the *EFFECTIVE SQL PARAMETER LIST*, and let *P* be the *PROGRAM* in an application of the General Rules of this Subclause.

2) Let *D* be the MD-extent of *AR*, let *ARC* be the cardinality of *D*, let *d* be the number of MD-axes in *D*, and let N_j, LO_j, and HI_j be the name, lower limit, and upper limit, respectively, of the *j*-th MD-axis in *D*, 1 (one) ≤ *j* ≤ *d*.

3) For all *j*, 1 (one) ≤ *j* ≤ *d*, let AE_j be the value of $HI_j - LO_j + 1$ and let PAE_j be the value of $AE_j \times AE_{j+1} \times \ldots \times AE_d$.

4) Let *EN* be the number of entries in *ESPL*.

5) Let ESP_i, 1 (one) ≤ *i* ≤ *EN*, be the *i*-th parameter in *ESPL*.

6) Let *FRN* be the number of result data items.

7) Let *PN* and *N* be the number of values in the static SQL argument list of *P*.

8) Depending on whether the language of *P* specifies ADA, C, COBOL, FORTRAN, M, PASCAL, or PLI, let the *operative data type correspondences table* be Table 2, "Data type correspondences for Ada", Table 3, "Data type correspondences for C", Table 4, "Data type correspondences for COBOL", Table 5,

"Data type correspondences for Fortran", Table 6, "Data type correspondences for M", Table 7, "Data type correspondences for Pascal", or Table 8, "Data type correspondences for PL/I", respectively. Refer to the two columns of the operative data type correspondences table as the *SQL data type column* and the *host data type column*.

9) For *i* varying from 1 (one) to *EN*, let PD_i be the *i*-th parameter of *P*, let PT_i be the <data type> of ESP_i, and let the host language data type DT_i of PD_i be the data type listed in the host data type column of the row in the operative data type correspondences table whose value in the SQL data type column is PT_i.

10) Let *E* be 0 (zero).

11) If the call type data item has a value of −1 (negative one, indicating *open call*), then:

    a) For *i* varying from 1 (one) through *EN*, the values of the parameters PD_i are set as follows.

       The General Rules of Subclause 9.4, "Passing a value from the SQL-server to a host language", are applied with the language of *P* as *LANGUAGE*, PT_i as *SQL TYPE*, and the value of ESP_i as *SQL VALUE*; let PD_i be the *HOST VALUE* returned from the application of those General Rules.

    b) *P* is executed.

    c) For *i* varying from 1 (one) through *EN*, the General Rules of Subclause 9.3, "Passing a value from a host language to the SQL-server", are applied with the language of *P* as *LANGUAGE*, PT_i as *SQL TYPE*, and the value of PD_i as *HOST VALUE*; let ESP_i be the *SQL VALUE* returned from the application of those General Rules.

12) Case:

    a) If the value of the exception data item is '00000' (corresponding to the completion condition *successful completion (00000)*) or the first 2 characters of the exception data item are '01' (corresponding to the completion condition *warning (01000)* with any subcondition), then set the call type data item to 0 (zero) (indicating *fetch call*).

    b) If the exception data item is '02000' (corresponding to the completion condition *no data (02000)*):

       i) If each PD_i ($(PN+FRN)+N+1) \le i \le ((PN+FRN)+N+FRN$) (that is, the SQL indicator arguments corresponding to the result data items), is negative, then set *AR* to the null value.

       ii) Set the call type data item to 1 (one) (indicating *close call*).

    c) Otherwise, set the call type data item to 1 (one) (indicating *close call*).

13) The following steps are applied repeatedly until the call type data item has a value other than 0 (zero) (corresponding to *fetch call*):

    a) For *i* ranging from 1 (one) to *EN*−2, the General Rules of Subclause 9.4, "Passing a value from the SQL-server to a host language", are applied with the language of *P* as *LANGUAGE*, PT_i as *SQL TYPE*, and the value of ESP_i as *SQL VALUE*; let PD_i be the *HOST VALUE* returned from the application of those General Rules.

    b) The value of PD_{EN-1} (that is, the save area data item) is set to the value returned in PD_{EN-1} by the prior execution of *P*.

    c) The value of PD_{EN} (that is, the call type data item) is set to 0 (zero).

    d) *P* is executed.

e) For *i* varying from 1 (one) through *EN*, the General Rules of Subclause 9.3, "Passing a value from a host language to the SQL-server", are applied with the language of *P* as *LANGUAGE*, PT_i as *SQL TYPE*, and the value of PD_i as *HOST VALUE*; let ESP_i be the *SQL VALUE* returned from the application of those General Rules.

f) Case:

i) If the exception data item is '00000' (corresponding to completion condition *successful completion (00000)*) or the first 2 characters of the exception data item are '01' (corresponding to completion condition *warning (01000)* with any subcondition), then:

1) Increment *E* by 1 (one).

2) If *E* > *ARC*, then an exception condition is raised: *data exception — MD-array element error (2203X)*.

3) Let *RM* be *E*. For *j*, 1 (one) ≤ *j* ≤ (*d*−1):

A) Let R_j be the value of CAST(((*RM* / PAE_{j+1}) + LO_j) AS INT).

B) Let *RM* be the value of MOD(*RM*, PAE_{j+1}).

4) Let R_d be the value of *RM*.

5) Let *Q* be the coordinate denoted by [R_1, ..., R_d].

6) Case:

A) If each PD_i, ((*PN*+*FRN*)+*N*+1) ≤ *i* ≤ ((*PN*+*FRN*)+*N*+*FRN*) (that is, the SQL indicator arguments corresponding to the result data items) is negative, then let the element of *AR* at coordinate *Q* be the null value.

B) If *FRN* is 1 (one), then let the element of *AR* at coordinate *Q* be the value of the result data item.

C) Otherwise:

I) Let RDI_i, 1 (one) ≤ *i* ≤ *FRN*, be the value of the *i*-th result data item.

II) Let the element of *AR* at coordinate *Q* be the value of the following <row value expression>:

ROW (RDI_1, ..., RDI_{FRN})

ii) If the exception data item is '02000' (corresponding to completion condition *no data (02000)*), then:

1) If the value of *E* is 0 (zero), then set *AR* to the null value.

2) Set the call type data item to 1 (one) (indicating *close call*).

iii) Otherwise, set the value of the call type data item to 1 (one) (indicating *close call*).

14) If the call type data item has a value of 1 (one) (indicating *close call*), then *P* is executed with parameters whose values are set as follows:

a) For *i* ranging from 1 (one) to *EN*−2, the General Rules of Subclause 9.4, "Passing a value from the SQL-server to a host language", are applied with the language of *P* as *LANGUAGE*, PT_i as *SQL TYPE*, and the value of ESP_i as *SQL VALUE*; let PD_i be the *HOST VALUE* returned from the application of those General Rules.

b) The value of PD_{EN-1} (that is, the save area data item) is set to the value returned in PD_{EN-1} by the prior execution of *P*.

c) The value of PD_{EN} (that is, the call type data item) is set to 1 (one).

d) *P* is executed.

15) Evaluation of the General Rules is terminated and control is returned to the invoking Subclause.

Conformance Rules

None.

10 Additional common elements

This Clause modifies Clause 10, "Additional common elements", in ISO/IEC 9075-2.

10.1 <md-extent alternative>

Function

Specify an MD-extent or a maximum MD-extent.

Format

```
<md-extent alternative> ::=
    <md-extent>
  | <md-array subset extent>

<md-extent> ::=
  <left bracket or trigraph> <md-axis list> <right bracket or trigraph>

<md-axis list> ::=
  <md-axis> [ { <comma> <md-axis> }... ]

<md-axis> ::=
  <md-axis name>
      <left paren> <md-axis lower limit> <colon> <md-axis upper limit> <right paren>

<md-axis lower limit> ::=
  <numeric value expression>

<md-axis upper limit> ::=
  <numeric value expression>
```

Syntax Rules

1) The maximum MD-extent specified by <md-extent alternative> is the maximum MD-extent specified by the immediately contained <md-extent> or <md-array subset extent>.

2) The declared types of <md-axis lower limit> and <md-axis upper limit> shall be exact numeric with scale 0 (zero).

3) If <md-extent> *EU* is specified, then:

   a) Let *d* be the number of <md-axis> simply contained in *EU*.

   b) The maximum number of <md-axis> is implementation-defined (IL203). *d* shall be less than this maximum value.

   c) Let MN_i, MLO_i, and MHI_i, 1 (one) ≤ i ≤ d, be the <md-axis name>, <md-axis lower limit>, and <md-axis upper limit>, respectively, of the i-th <md-axis> immediately contained in the <md-axis list> immediately contained in <md-extent>.

      i) The declared types of MLO_i and MHI_i shall be exact numeric with scale 0 (zero).

ii) For all i and j, 1 (one) ≤ $i, j \le d$, $i \ne j$, MN_i shall not be equivalent to MN_j.

d) Let *MD* be the maximum MD-extent denoted by [MN_1 (NULL : NULL), ..., MN_d (NULL : NULL)]. *MD* is the maximum MD-extent specified by *EU*.

4) If <md-array subset extent> *AE* is specified, then let *AVE* be the <md-array value expression> immediately contained in *AE* and let *MD* be the maximum MD-extent of *AVE*. *MD* is the maximum MD-extent specified by *AE*.

Access Rules

None.

General Rules

1) The MD-extent defined by <md-extent alternative> is the MD-extent defined by the immediately contained <md-extent> or <md-array subset extent>.

2) If <md-extent> *EU* is specified, then:

a) For all i, 1 (one) ≤ $i \le d$:

i) Let A_i be the i-th <md-axis> simply contained in *EU*.

ii) Let N_i be the value of the <md-axis name> immediately contained in A_i.

iii) Case:

1) If the value of the <md-axis lower limit> immediately contained in A_i is the null value, then an exception condition is raised: *data exception — MD-array null limit in MD-extent (2203K)*.

2) Otherwise, let LO_i be the value of the <md-axis lower limit> immediately contained in A_i.

iv) Case:

1) If the value of the <md-axis upper limit> immediately contained in A_i is the null value, then an exception condition is raised: *data exception — MD-array null limit in MD-extent (2203K)*.

2) Otherwise, let HI_i be the value of the <md-axis upper limit> immediately contained in A_i.

v) If LO_i is greater than HI_i, then an exception condition is raised: *data exception — MD-array lower limit greater than upper limit (2203U)*.

vi) If LO_i is less than MLO_i or HI_i is greater than MHI_i, then an exception condition is raised: *data exception — MD-array limit in MD-extent out of bounds (22044)*.

b) Let *D* be the MD-extent denoted by [N_1 (LO_1 : HI_1), ..., N_d (LO_d : HI_d)]. *D* is the MD-extent specified by *EU*.

3) If <md-array subset extent> *AE* is specified, then let *D* be the MD-extent of the value of *AVE*. *D* is the MD-extent specified by *AE*.

Conformance Rules

1) Without Feature A010, “Multi-dimensional array support”, conforming SQL language shall not contain an <md-extent alternative>.

10.2 <md-array md-axis>

Function

Specify an MD-axis as a name or index.

Format

```
<md-array md-axis> ::=
  <md-array value expression> <comma> <md-axis identifier>

<md-axis identifier> ::=
    <unsigned integer>
  | <md-axis name>
```

Syntax Rules

1) Let *MDT* be the declared type of the <md-array value expression>, let *d* be the MD-dimension of *MDT*, and let N_i, 1 (one) ≤ *i* ≤ *d*, be the name of the *i*-th MD-axis in the maximum MD-extent of *MDT*.

   Case:

   a) If the <md-axis identifier> *AI* immediately contained in an <md-array md-axis> is an <unsigned integer>, then the value of *AI* shall be greater than 0 (zero), and less than or equal to *d*. Let *AD* be the value of the <unsigned integer>.

   b) Otherwise, the immediately contained <md-axis name> shall be equivalent to N_i, for some *i*, 1 (one) ≤ *i* ≤ *d*. Let *AD* = *i*.

2) *AD* is the *MD-axis index* specified by <md-array md-axis>.

Access Rules

None.

General Rules

None.

Conformance Rules

1) Without Feature A010, "Multi-dimensional array support", conforming SQL language shall not contain an <md-array md-axis>.

11 Schema definition and manipulation

This Clause modifies Clause 11, "Schema definition and manipulation", in ISO/IEC 9075-2.

11.1 <column definition>

This Subclause modifies Subclause 11.4, "<column definition>", in ISO/IEC 9075-2.

Function

Define a column of a base table.

Format

No additional Format items.

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

1) Insert after the last CR: Without Feature A016, "Persistent MDA values", conforming SQL language shall not contain a <column definition> whose declared type is based on an MD-array type.

11.2 <view definition>

This Subclause modifies Subclause 11.32, "<view definition>", in ISO/IEC 9075-2.

Function

Define a viewed table.

Format

No additional Format items.

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

1) Insert after the last CR: Without Feature A016, "Persistent MDA values", conforming SQL language shall not contain a <view definition> that defines a column whose declared type is based on an MD-array type.

11.3 <user-defined type definition>

This Subclause modifies Subclause 11.51, "<user-defined type definition>", in ISO/IEC 9075-2.

Function

Define a user-defined type.

Format

No additional Format items.

Syntax Rules

1) Augment SR 7)a) by adding "*PSDT* shall not be an MD-array type. " at the end of the lead text of the rule.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

11.4 <SQL-invoked routine>

This Subclause modifies Subclause 11.60, "<SQL-invoked routine>", in ISO/IEC 9075-2.

Function

Define an SQL-invoked routine.

Format

No additional Format items.

Syntax Rules

1) Insert before SR 9)p): An *MD-array-returning external function* is an SQL-invoked function that is an external routine for which exactly one of the following is true:

   a) A <result cast from type> is specified that does not contain a <locator indication>, but simply contains an <md-array type>.

   b) A <result cast from type> is not specified and <returns data type> does not contain a <locator indication>, but simply contains an <md-array type>.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

12 SQL-client modules

This Clause modifies Clause 13, "SQL-client modules", in ISO/IEC 9075-2.

12.1 <externally-invoked procedure>

This Subclause modifies Subclause 13.3, "<externally-invoked procedure>", in ISO/IEC 9075-2.

Function

Define an externally-invoked procedure.

Format

No additional Format items.

Syntax Rules

1) Insert into SR 4)a), before the 4)a)v) list item:

   a) MD-array type;

2) Insert after SR 4)a): If *T* is an MD-array type, then the host parameter identified by <host parameter name> is called an *MD-array locator parameter*.

3) Insert into SR 10)e) after

```
package SQLSTATE_CODES is
```

the code:

```
DATA_EXCEPTION_NO_SUBCLASS:
  constant SQLSTATE_TYPE := "22000";
DATA_EXCEPTION_MD_ARRAY_NULL_LIMIT_IN_SUBSET:
  constant SQLSTATE_TYPE := "2203J";
DATA_EXCEPTION_MD_ARRAY_NULL_LIMIT_IN_MD_EXTENT:
  constant SQLSTATE_TYPE := "2203K";
DATA_EXCEPTION_MD_ARRAY_SUBSET_NOT_WITHIN_MD_EXTENT:
  constant SQLSTATE_TYPE := "2203L";
DATA_EXCEPTION_MD_ARRAY_DUPLICATE_COORDINATE_IN_QUERY_CONSTRUCTOR:
  constant SQLSTATE_TYPE := "2203M";
DATA_EXCEPTION_MD_ARRAY_NULL_COORDINATE_IN_QUERY_CONSTRUCTOR:
  constant SQLSTATE_TYPE := "2203N";
DATA_EXCEPTION_MD_ARRAY_COORDINATE_NOT_WITHIN_SPECIFIED_MD_EXTENT:
  constant SQLSTATE_TYPE := "2203P";
DATA_EXCEPTION_MD_ARRAY_SOURCE_MD_EXTENT_NOT_STRICTLY_WITHIN_MAXIMUM_TARGET_MD_EXTENT:
  constant SQLSTATE_TYPE := "2203Q";
DATA_EXCEPTION_MD_ARRAY_OPERANDS_WITH_NON_MATCHING_MD_EXTENTS:
  constant SQLSTATE_TYPE := "2203R";
DATA_EXCEPTION_MD_ARRAY_INVALID_MD_AXIS:
  constant SQLSTATE_TYPE := "2203T";
DATA_EXCEPTION_MD_ARRAY_LOWER_LIMIT_GREATER_THAN_UPPER_LIMIT:
  constant SQLSTATE_TYPE := "2203U";
DATA_EXCEPTION_MD_ARRAY_ELEMENT_ERROR:
```

```
  constant SQLSTATE_TYPE := "2203X";
DATA_EXCEPTION_MD_ARRAY_DECODING_ERROR:
  constant SQLSTATE_TYPE := "2203Y";
DATA_EXCEPTION_MD_ARRAY_ENCODING_ERROR:
  constant SQLSTATE_TYPE := "2203Z";
DATA_EXCEPTION_MD_ARRAY_ELEMENT_REFERENCE_NOT_WITHIN_MD_EXTENT:
  constant SQLSTATE_TYPE := "22040";
DATA_EXCEPTION_MD_ARRAY_NULL_VALUE_IN_MD_ARRAY_TARGET:
  constant SQLSTATE_TYPE := "22041";
DATA_EXCEPTION_MD_ARRAY_SOURCE_MD_EXTENT_NOT_STRICTLY_WITHIN_TARGET_MD_EXTENT:
  constant SQLSTATE_TYPE := "22042";
DATA_EXCEPTION_MD_ARRAY_TARGET_MD_EXTENT_NOT_STRICTLY_WITHIN_MAXIMUM_MD_EXTENT:
  constant SQLSTATE_TYPE := "22043";
DATA_EXCEPTION_MD_ARRAY_LIMIT_IN_MD_EXTENT_OUT_OF_BOUNDS:
  constant SQLSTATE_TYPE := "22044";
```

The text of the Ada library unit package Interfaces.SQL is also available from the ISO website as a "digital artifact". See https://standards.iso.org/iso-iec/9075/-15/ed-2/en/ to download digital artifacts for this document. To download the library unit package, select the file named ISO_IEC_9075-15(E)_MDA-Interfaces.SQL.ada.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

12.2 Data type correspondences

This Subclause modifies Subclause 13.5, "Data type correspondences", in ISO/IEC 9075-2.

Function

Specify the data type correspondences for SQL data types and host language types.

Tables

Insert into Table 21, "Data type correspondences for Ada" the rows of Table 2, "Data type correspondences for Ada".

Table 2 — Data type correspondences for Ada

SQL Data Type	Ada Data Type
MDARRAY	*None*

Insert into Table 22, "Data type correspondences for C" the rows of Table 3, "Data type correspondences for C".

Table 3 — Data type correspondences for C

SQL Data Type	C Data Type
MDARRAY	*None*

Insert into Table 23, "Data type correspondences for COBOL" the rows of Table 4, "Data type correspondences for COBOL".

Table 4 — Data type correspondences for COBOL

SQL Data Type	COBOL Data Type
MDARRAY	*None*

Insert into Table 24, "Data type correspondences for Fortran" the rows of Table 5, "Data type correspondences for Fortran".

Table 5 — Data type correspondences for Fortran

SQL Data Type	Fortran Data Type
MDARRAY	*None*

Insert into Table 25, "Data type correspondences for M" the rows of Table 6, "Data type correspondences for M".

Table 6 — Data type correspondences for M

SQL Data Type	M Data Type
MDARRAY	*None*

Insert into Table 26, "Data type correspondences for Pascal" the rows of Table 7, "Data type correspondences for Pascal".

Table 7 — Data type correspondences for Pascal

SQL Data Type	Pascal Data Type
MDARRAY	*None*

Insert into Table 27, "Data type correspondences for PL/I" the rows of Table 8, "Data type correspondences for PL/I".

Table 8 — Data type correspondences for PL/I

SQL Data Type	PL/I Data Type
MDARRAY	*None*

Conformance Rules

None.

13 Data manipulation

This Clause modifies Clause 14, "Data manipulation", in ISO/IEC 9075-2.

13.1 <set clause list>

This Subclause modifies Subclause 14.15, "<set clause list>", in ISO/IEC 9075-2.

Function

Specify a list of updates.

Format

```
<update target> ::=
    !! All alternatives from ISO/IEC 9075-2
  | <object column>
        <left bracket or trigraph> <md-axis slice list> <right bracket or trigraph>
  | <object column>
        <left bracket or trigraph> <md-axis subset list> <right bracket or trigraph>
```

Syntax Rules

1) Insert after SR 11)a): If the <update target> immediately contains an <md-axis slice list> *ASL*, then the declared type *MDT* of the column *C* identified by the <object column> *OC* shall be an MD-array type.

   a) The Syntax Rules of Subclause 9.17, "Canonicalize MD-array element reference", are applied with *MDT* as *MDARRAY TYPE* and *ASL* as *MDAXIS SLICE LIST*; let *CASL* be the *CANONICAL SLICE COORDINATE* returned from the application of those Syntax Rules.

   b) Let *EDT* be the element type of *MDT*.

   c) The Syntax Rules of Subclause 9.2, "Store assignment", in ISO/IEC 9075-2, are applied with a temporary site whose declared type is *EDT* as *TARGET* and the <update source> of the <set clause> as *VALUE*.

2) Insert after SR 11)a): If the <update target> immediately contains an<md-axis subset list> *ASL*, then the declared type *MDT* of the column of *T* identified by the <object column> *OC* shall be an MD-array type.

   The Syntax Rules of Subclause 9.16, "MD-array subset", are applied with *MDT* as *MDARRAY TYPE* and *ASL* as *MDAXIS SUBSET LIST*; let *TMD* be the *TARGET MAXIMUM MDEXTENT* and let *EQ* be the *EQUIVALENCE* returned from the application of those Syntax Rules. Let *t* be the number of MD-axes in *TMD*.

   Case:

   a) If *EQ* is not a zero-length character string, then <update target> is equivalent to *OC EQ*.

   b) Otherwise:

      i) Let *EDT* be the element type of *MDT*.

ii) The Syntax Rules of Subclause 9.2, "Store assignment", in ISO/IEC 9075-2, are applied with a temporary site whose declared type is an MD-array type with element type *EDT* and maximum MD-extend *TMD* as *TARGET* and the <update source> of the <set clause> as *VALUE*.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

1) Insert after the last CR: Without Feature A010, "Multi-dimensional array support", conforming SQL language shall not contain an <update target> that immediately contains an <md-axis slice list> or an <md-axis subset list>.

14 Additional data manipulation rules

This Clause modifies Clause 15, "Additional data manipulation rules", in ISO/IEC 9075-2.

14.1 Evaluating a <set clause list>

This Subclause modifies Subclause 15.5, "Evaluating a <set clause list>", in ISO/IEC 9075-2.

Function

Specify a list of updates.

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

1) The General Rules of Subclause 8.6, "Handler execution", in ISO/IEC 9075-4, are applied with *EDT* as *MOST APPROPRIATE HANDLER*.

2) Insert before GR 3)b)ii): If the *i*-th <set clause> contains an <update target> *AS* that immediately contains an <md-axis slice list> or <md-axis subset list>, then:

   a) Let *ASL* be the <md-axis slice list> or <md-axis subset list> immediately contained in *AS*.

   b) If the value of *C* is the null value, then an exception condition is raised: *data exception — MD-array null value in MD-array target (22041)*.

   c) Let *AV* be the value of *C*. Let *SV* be the *i*-th update value.

   d) Let *D* be the MD-extent of *AV*, let *d* be the number of MD-axes in *D*, and let *MD* be the maximum MD-extent of *AV*.

   e) Case:

      i) If *ASL* is an <md-axis slice list>, then:

         1) Let P_j be the value of the *j*-th element of *CASL*, 1 (one) $\leq j \leq d$.

         2) Let MN_j be the name of the *j*-th axis in *MD*, 1 (one) $\leq j \leq d$.

         3) Let *SD* be the MD-extent denoted by [MN_1(P_1 : P_1), ..., MN_d(P_d : P_d)].

         4) Let *VD* be *SD*.

      ii) Otherwise:

1) The General Rules of Subclause 9.16, "MD-array subset", are applied with *AV* as *MDARRAY VALUE* and *ASL* as *MDAXIS SUBSET LIST*; let *SD* be the *SOURCE MDEXTENT*, let *TD* be the *TARGET MDEXTENT*, and let *SLICEDAXES* be the *SLICED MDAXES* returned from the application of those General Rules.

2) Let *VD* be the MD-extent of *SV*.

3) If *VD* is not strictly within *TMD*, then an exception condition is raised: *data exception — MD-array source MD-extent not strictly within target MD-extent (22042)*.

f) If *SD* is not strictly within *MD*, then an exception condition is raised: *data exception — MD-array target MD-extent not strictly within maximum MD-extent (22043)*.

g) Let *UD* be the union of *D* and *SD*.

h) *AV* is replaced by an MD-array *A* with element type *EDT* and MD-extent *UD* derived as follows.

For every coordinate *R* within *UD*:

i) Case:

1) If *ASL* is an <md-axis slice list>, then let *Q* be *R*.

2) Otherwise:

A) Let RP_j be the *j*-th element of *R*, 1 (one) $\leq j \leq d$.

B) Let *j* be 1 (one).

C) For all *k*, 1 (one) $\leq k \leq d$, such that *k* is not an element of *SLICEDAXES*:

I) Let TP_j be RP_k.

II) *j* is increased by 1 (one).

D) Let *Q* be the coordinate denoted by [TP_1, ..., TP_t].

NOTE 5 — There are *t* non-*SLICEDAXES*.

ii) Case:

1) If *R* is within *D* and *Q* is not within *VD*, then the element in *A* at coordinate *R* is the value of the element in *AV* at coordinate *R*.

2) If *R* is not within *D* and *Q* is not within *VD*, then the element in *A* at coordinate *R* is the null value.

3) Otherwise:

A) Case:

I) If *ASL* is an <md-axis slice list>, then let *V* be *SV*.

II) Otherwise, let *V* be the element of *SV* at coordinate *Q*.

B) The General Rules of Subclause 9.2, "Store assignment", in ISO/IEC 9075-2, are applied with the element of *A* at coordinate *R* as *TARGET* and *V* as *VALUE*.

Conformance Rules

No additional Conformance Rules.

15 Dynamic SQL

This Clause modifies Clause 20, "Dynamic SQL", in ISO/IEC 9075-2.

15.1 Description of SQL descriptor areas

This Subclause modifies Subclause 20.1, "Description of SQL descriptor areas", in ISO/IEC 9075-2.

Function

Specify the identifiers, data types, and codes used in SQL item descriptor areas.

Syntax Rules

1) Insert after SR 6)q): TYPE indicates MDARRAY or MDARRAY LOCATOR, the value *d* of MDDIMENSION is a valid value for the MD-dimension of an MD-array, there are exactly *d* + 1 (one) immediately subordinate descriptor areas of *IDA*, and those SQL item descriptor areas are valid.

2) Insert after SR 7)v): TYPE indicates MDARRAY or MDARRAY LOCATOR, and *T* is an MD-array type with MD-dimension *d*, where *d* is the value of MDDIMENSION, and the data type of the element type of *T* matches the data type specified by the *e*-th immediately subordinate descriptor area of *IDA*, where *e* = *d* + 1 (one). The name, lower limit, and upper limit of the *i*-th MD-axis match the value of NAME, MDAXIS_LOW, and MDAXIS_HIGH specified by the *i*-th immediately subordinate descriptor area of *IDA*.

3) Insert into Table 28, "Data types of <key word>s used in the header of SQL descriptor areas" the rows of Table 9, "Data types of <key word>s used in SQL item descriptor areas".

Table 9 — Data types of <key word>s used in SQL item descriptor areas

<key word>	Data Type
MDDIMENSION	exact numeric with scale 0 (zero)
MDAXIS_LOW	exact numeric with scale 0 (zero)
MDAXIS_HIGH	exact numeric with scale 0 (zero)

Access Rules

No additional Access Rules.

General Rules

1) Insert into Table 30, "Codes used for SQL data types in Dynamic SQL" the rows of Table 10, "Codes used for SQL data types in Dynamic SQL".

Table 10 — Codes used for SQL data types in Dynamic SQL

Data Type	Code
MDARRAY	52
MDARRAY LOCATOR	53

Conformance Rules

No additional Conformance Rules.

15.2 <get descriptor statement>

This Subclause modifies Subclause 20.4, "<get descriptor statement>", in ISO/IEC 9075-2.

Function

Get information from an SQL descriptor area.

Format

```
<descriptor item name> ::=
    !! All alternatives from ISO/IEC 9075-2
  | MDDIMENSION
  | MDAXIS_LOW
  | MDAXIS_HIGH
```

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

15.3 <describe statement>

This Subclause modifies Subclause 20.10, "<describe statement>", in ISO/IEC 9075-2.

Function

Obtain information about the <select list> columns or <dynamic parameter specification>s contained in a prepared statement or about the columns of the result set associated with a cursor.

Format

No additional Format items.

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

1) Insert after GR 7)d)xi): If TYPE indicates MDARRAY, then MDDIMENSION is set to the MD-dimension *d* of the MD-array *A*.

2) Insert after GR 7)d)xi): If WITH NESTING is specified, then:

   a) Let *D* be the MD-extent of *A*, let *d* be the number of MD-axes in *D*, and let N_i, LO_i, and HI_i, be the name, lower limit, and upper limit, respectively, of the *i*-th MD-axis in *D*, 1 (one) ≤ *i* ≤ *d*.

   b) For all *i*, 1 (one) ≤ *i* ≤ *d*, in the *i*-th immediately subordinate descriptor area, NAME is set to N_i, MDAXIS_LOW is set to LO_i and MDAXIS_HIGH is set to HI_i.

   c) The (*d* + 1)-th immediately subordinate descriptor area is set to describe the element type of *A*.

Conformance Rules

No additional Conformance Rules.

16 Embedded SQL

This Clause modifies Clause 21, "Embedded SQL", in ISO/IEC 9075-2.

16.1 <embedded SQL Ada program>

This Subclause modifies Subclause 21.3, "<embedded SQL Ada program>", in ISO/IEC 9075-2.

Function

Specify an <embedded SQL Ada program>.

Format

```
<Ada derived type specification> ::=
    !! All alternatives from ISO/IEC 9075-2
  | <Ada md-array locator variable>

<Ada md-array locator variable> ::=
  SQL TYPE IS <md-array type> AS LOCATOR
```

Syntax Rules

1) Insert after SR 4)c)xi): If *ATS* immediately contains an <Ada md-array locator variable> *AMDLV*, then the <host parameter data type> of *HV* is *AT* AS LOCATOR, where *AT* is the <md-array type> immediately contained in *AMDLV*.

2) Insert after SR 5)l): The syntax

   ```
   SQL TYPE IS <md-array type> AS LOCATOR
   ```

   shall be replaced by

   ```
   Interfaces.SQL.INT
   ```

   in any <Ada md-array locator variable>. The host variable defined by <Ada md-array locator variable> is called an *MD-array locator variable*. The data type identified by <md-array type> is called the *associated MD-array type* of the host variable.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

1) Insert after the last CR: Without Feature A010, “Multi-dimensional array support”, conforming SQL language shall not contain an <Ada md-array locator variable>.

16.2 <embedded SQL C program>

This Subclause modifies Subclause 21.4, "<embedded SQL C program>", in ISO/IEC 9075-2.

Function

Specify an <embedded SQL C program>.

Format

```
<C derived variable> ::=
    !! All alternatives from ISO/IEC 9075-2
  | <C md-array locator variable>

<C md-array locator variable> ::=
  SQL TYPE IS <md-array type> AS LOCATOR
      <C host identifier> [ <C initial value> ]
      [ { <comma> <C host identifier> [
      <C initial value> ] }... ]
```

Syntax Rules

1) Insert after SR 4)c)vii): If *CVS* simply contains a <C md-array locator variable> *CMDLV*, then the <host parameter data type> of *HV* is *AT* AS LOCATOR, where *AT* is the <md-array type> immediately contained in *CMDLV*.

2) Insert after SR 5)o): The syntax

   ```
   SQL TYPE IS <md-array type> AS LOCATOR
   ```

   shall be replaced by

   ```
   unsigned long
   ```

   in any <C md-array locator variable>. The host variable defined by <C md-array locator variable> is called an *MD-array locator variable*. The data type identified by <md-array type> is called the *associated MD-array type* of the host variable.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

1) Insert after the last CR: Without Feature A010, "Multi-dimensional array support", conforming SQL language shall not contain a <C md-array locator variable>.

16.3 <embedded SQL COBOL program>

This Subclause modifies Subclause 21.5, "<embedded SQL COBOL program>", in ISO/IEC 9075-2.

Function

Specify an <embedded SQL COBOL program>.

Format

```
<COBOL derived type specification> ::=
    !! All alternatives from ISO/IEC 9075-2
  | <COBOL md-array locator variable>

<COBOL md-array locator variable> ::=
  [ USAGE [ IS ] ] SQL TYPE IS <md-array type> AS LOCATOR
```

Syntax Rules

1) Insert after SR 4)d)x): If *CTS* immediately contains a <COBOL md-array locator variable> *CMDLV*, then the <host parameter data type> of *HV* is *AT* AS LOCATOR, where *AT* is the <md-array type> immediately contained in *CMDLV*.

2) Insert after SR 5)m): The syntax

   ```
   SQL TYPE IS <md-array type> AS LOCATOR
   ```

   shall be replaced by

   ```
   PIC S9(9) USAGE IS BINARY
   ```

   in any <COBOL md-array locator variable>. The host variable defined by <COBOL md-array locator variable> is called an *MD-array locator variable*. The data type identified by <md-array type> is called the *associated MD-array type* of the host variable.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

1) Insert after the last CR: Without Feature A010, "Multi-dimensional array support", conforming SQL language shall not contain a <COBOL md-array locator variable>.

16.4 <embedded SQL Fortran program>

This Subclause modifies Subclause 21.6, "<embedded SQL Fortran program>", in ISO/IEC 9075-2.

Function

Specify an <embedded SQL Fortran program>.

Format

```
<Fortran derived type specification> ::=
    !! All alternatives from ISO/IEC 9075-2
  | <Fortran md-array locator variable>

<Fortran md-array locator variable> ::=
  SQL TYPE IS <md-array type> AS LOCATOR
```

Syntax Rules

1) Insert after SR 5)g)xi): If *FTS* immediately contains a <Fortran md-array locator variable> *FMDLV*, then the <host parameter data type> of *HV* is *AT* AS LOCATOR, where *AT* is the <md-array type> immediately contained in *FMDLV*.

2) Insert after SR 6)l): The syntax

   ```
   SQL TYPE IS <md-array type> AS LOCATOR
   ```

   shall be replaced by

   ```
   INTEGER
   ```

   in any <Fortran md-array locator variable>. The host variable defined by <Fortran md-array locator variable> is called an *MD-array locator variable*. The data type identified by <md-array type> is called the *associated MD-array type* of the host variable.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

1) Insert after the last CR: Without Feature A010, "Multi-dimensional array support", conforming SQL language shall not contain a <Fortran md-array locator variable>.

16.5 <embedded SQL MUMPS program>

This Subclause modifies Subclause 21.7, "<embedded SQL MUMPS program>", in ISO/IEC 9075-2.

Function

Specify an <embedded SQL MUMPS program>.

Format

```
<MUMPS derived type specification> ::=
    !! All alternatives from ISO/IEC 9075-2
  | <MUMPS md-array locator variable>

<MUMPS md-array locator variable> ::=
  SQL TYPE IS <md-array type> AS LOCATOR
```

Syntax Rules

1) Insert after SR 4)c)v): If *MVD* immediately contains a <MUMPS md-array locator variable> *MMDLV*, then let *AT* be the <md-array type> immediately contained in *MMDLV*. The <host parameter data type> of *HV* is *AT* AS LOCATOR. The data type identified by *AT* is called the *associated MD-array type* of *HV*. *HV* is called an *MD-array locator variable*.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

1) Insert after the last CR: Without Feature A010, "Multi-dimensional array support", conforming SQL language shall not contain a <MUMPS md-array locator variable>.

16.6 <embedded SQL PL/I program>

This Subclause modifies Subclause 21.9, "<embedded SQL PL/I program>", in ISO/IEC 9075-2.

Function

Specify an <embedded SQL PL/I program>.

Format

```
<PL/I derived type specification> ::=
    !! All alternatives from ISO/IEC 9075-2
  | <PL/I md-array locator variable>

<PL/I md-array locator variable> ::=
  SQL TYPE IS <md-array type> AS LOCATOR
```

Syntax Rules

1) Insert after SR 4)e)xi): If *PTS* immediately contains a <PL/I md-array locator variable> *PMDLV*, then the <host parameter data type> of *HV* is *AT* AS LOCATOR, where *AT* is the <md-array type> immediately contained in *PMDLV*.

2) Insert after SR 5)l): The syntax

   ```
   SQL TYPE IS <md-array type> AS LOCATOR
   ```

   shall be replaced by

   ```
   FIXED BINARY(31)
   ```

   in any <PL/I md-array locator variable>. The host variable defined by <PL/I md-array locator variable> is called an *MD-array locator variable*. The data type identified by <md-array type> is called the *associated MD-array type* of the host variable.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

1) Insert after the last CR: Without Feature A010, "Multi-dimensional array support", conforming SQL language shall not contain a <PL/I md-array locator variable>.

