

ISO/IEC JTC 1/SC 32 N 0651

Date: 2001-05-29

REPLACES: --

<p style="text-align: center;">ISO/IEC JTC 1/SC 32</p> <p style="text-align: center;">Data Management and Interchange</p> <p style="text-align: center;">Secretariat: United States of America (ANSI)</p> <p style="text-align: center;">Administered by Pacific Northwest National Laboratory on behalf of ANSI</p>
--

DOCUMENT TYPE	Working Draft Text
TITLE	ISO/IEC WD 9075-7:200x Information technology - Database languages - SQL - Part 7: Temporal (SQL/Temporal)
SOURCE	Jim Melton (Editor)
PROJECT NUMBER	1.32.03.05.07.00
STATUS	The document presented here is submitted as a candidate base document
REFERENCES	
ACTION ID.	FYI
REQUESTED ACTION	
DUE DATE	
Number of Pages	129
LANGUAGE USED	English
DISTRIBUTION	P & L Members SC Chair WG Conveners and Secretaries

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

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

Telephone: +1 703 575 2114; Facsimile: +1 703 671 9180; E-mail: MannD@battelle.org

available from the JTC 1/SC 32 WebSite <http://www.jtc1sc32.org/>

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

WG3:YYJ-007

H2-2001-144

May, 2001

ISO
International Organization for Standardization
ANSI
American National Standards Institute

ANSI TC NCITS H2
ISO/IEC JTC 1/SC 32/WG 3
Database

Title: (ISO-ANSI Working Draft) Temporal (SQL/Temporal)

Author: Jim Melton (Editor)

References:

- 1) WG3:YYJ-003 = H2-2001-140, *CD 9075-1 (SQL/Framework)*, June, 2001
- 2) WG3:YYJ-004 = H2-2001-141, *CD 9075-2 (SQL/Foundation)*, June, 2001
- 3) WG3:YYJ-005 = H2-2001-142, *CD 9075-3 (SQL/CLI)*, June, 2001
- 4) WG3:YYJ-006 = H2-2001-143, *CD 9075-4 (SQL/PSM)*, June, 2001
- 5) WG3:YYJ-007 = H2-2001-144, *(ISO-ANSI Working Draft) Temporal (SQL/Temporal)*, May, 2001
- 6) WG3:YYJ-008 = H2-2001-145, *CD 9075-9 (SQL/MED)*, June, 2001
- 7) WG3:YYJ-009 = H2-2001-146, *CD 9075-10 (SQL/OLB)*, June, 2001
- 8) WG3:YYJ-010 = H2-2001-147, *CD 9075-11 (SQL/Schemata)*, June, 2001
- 9) WG3:YYJ-011 = H2-2001-148, *FCD 9075-13 (SQL/JRT)*, April, 2001
- 10) WG3:YYJ-012 = H2-2001-149, *WD 9075-14 (SQL/XML)*, May, 2001

ISO/IEC JTC 1/SC 32

Date: 2001-05-26

[ANSI/]ISO/IEC 9075-7:200x(E)

ISO/IEC JTC 1/SC 32/WG 3

Secretariat: United States of America (ANSI)

**Information technology — Database languages — SQL — Part 7: Temporal
(SQL/Temporal)**

Technologies de l'information — Langages de base de donnée — SQL — Partie 7: Temporel (SQL/Temporel)

Document type: International standard
Document subtype:
Document stage: (20) Working Draft
Document language: E

Copyright notice

This ISO document is a Draft International Standard and is copyright-protected by ISO. Except as permitted under the applicable laws of the user's country, neither this ISO draft nor any extract from it may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, photocopying, recording, or otherwise, without prior written permission being secured.

Requests for permission to reproduce should be addressed to ISO at the address below or ISO's member body in the country of the requester.

*Copyright Manager
ISO Central Secretariat
1 rue de Varembé
1211 Geneva 20 Switzerland
tel. +41 22 749 0111
fax +41 22 734 1079
internet: iso@iso.ch*

Reproduction may be subject to royalty payments or a licensing agreement.

Violators may be prosecuted.

Contents	Page
Foreword	vi
Introduction	ix
1 Scope	1
2 Normative references	3
3 Definitions, notations, and conventions	5
3.1 Definitions	5
3.2 Notations	6
3.3 Conventions	6
3.3.1 Relationships to other parts of ISO/IEC 9075	6
3.3.1.1 New and modified Clauses, Subclauses, and Annexes	6
3.3.1.2 New and modified Format items	7
3.3.1.3 New and modified paragraphs and rules	7
3.3.1.4 New and modified tables	8
3.3.1.5 Clause, Subclause, and Table relationships	8
3.4 Object identifier for Database Language SQL	14
4 Concepts	15
4.1 Introduction to SQL/Temporal	15
4.2 Period data type	15
4.2.1 Operations involving periods	16
4.2.1.1 Functions that operate on periods and return periods	17
4.2.1.2 Functions that operate on periods and return values of element type	17
4.2.1.3 Other operations involving period elements	17
4.2.1.4 Other operations involving periods	17
4.2.2 Period type conversions and mixing of data types	17
4.2.3 Period predicates	18
4.3 Tables	18
4.3.1 Operations involving tables	18
4.3.1.1 Operations involving tables with period columns	18
4.4 Standard programming languages	19
4.5 Miscellaneous characteristics	19
4.5.1 Statement attributes	19

5	Lexical elements	21
5.1	<token> and <separator>	21
5.2	<literal>	23
6	Scalar expressions	25
6.1	<data type>	25
6.2	<set function specification>	26
6.3	<cast specification>	27
6.4	<value expression>	30
6.5	<period value expression>	33
6.6	<set or multiset value expression>	37
6.7	<period set value function>	38
6.8	<period value constructor>	40
7	Query expressions	43
7.1	<normalized query specification>	43
7.2	<query expression>	45
7.3	<expanding clause>	47
8	Predicates	51
8.1	<predicate>	51
8.2	<comparison predicate>	52
8.3	<overlaps predicate>	54
8.4	<period predicate>	56
9	Expansion and compression	59
9.1	Expansion of a table	59
9.2	Compression of a table	60
10	Additional common rules	61
10.1	Retrieval assignment	61
10.2	Store assignment	62
10.3	Set operation result data types	63
11	Schema definition and manipulation	65
11.1	<default clause>	65
12	SQL-client modules	68
12.1	Data type correspondences	68
13	Dynamic SQL	71
13.1	Description of SQL item descriptor areas	71
13.2	<get descriptor statement>	73
13.3	<set descriptor statement>	74
13.4	<prepare statement>	75
13.5	<describe statement>	76

14	Call-Level Interface specifications	77
14.1	Implicit EXECUTE USING and OPEN USING clauses	77
14.2	Description of CLI item descriptor areas	78
14.3	SQL/CLI data type correspondences	79
15	SQL/CLI routines	81
15.1	GetData	81
15.2	GetParamData	82
15.3	SetDescField	83
16	Information Schema	85
16.1	ELEMENT_TYPES view	85
17	Definition Schema	87
17.1	DATA_TYPE_DESCRIPTOR base table	87
17.2	ELEMENT_TYPES base table	89
18	Status codes	91
18.1	SQLSTATE	91
19	Conformance	93
Annex A	Implementation-defined elements	95
Annex B	Implementation-dependent elements	97
Annex C	Deprecated features	99
Annex D	Incompatibilities with ISO/IEC 9075:1999	101
Annex E	Typical header files	104
E.1	C header file SQLCLI.H	104
E.2	COBOL library item SQLCLI	104
Annex F	SQL Feature Taxonomy	105
Index		Index1

TABLES

Tables	Page
1 Clause, Subclause, and Table relationships	8
2 Data type correspondences for Ada	68
3 Data type correspondences for C	68
4 Data type correspondences for COBOL	68
5 Data type correspondences for Fortran	68
6 Data type correspondences for MUMPS	69
7 Data type correspondences for Pascal	69
8 Data type correspondences for PL/I	69
9 Codes used for SQL data types in Dynamic SQL	72
10 Codes associated with datetime and period data types in Dynamic SQL	72
11 Codes used for implementation data types in SQL/CLI	78
12 SQL/CLI data type correspondences for Ada	79
13 SQL/CLI data type correspondences for C	79
14 SQL/CLI data type correspondences for COBOL	79
15 SQL/CLI data type correspondences for Fortran	79
16 SQL/CLI data type correspondences for MUMPS	79
17 SQL/CLI data type correspondences for Pascal	79
18 SQL/CLI data type correspondences for PL/I	80
19 SQLSTATE class and subclass values	91

Foreword

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

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

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

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

Attention is drawn to the possibility that some of the elements of this International Standard may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

International Standard ISO/IEC 9075-7 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 32, *Data management and interchange*.

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

- Part 1: Framework (SQL/Framework)
- Part 2: Foundation (SQL/Foundation)
- Part 3: Call-Level Interface (SQL/CLI)
- Part 4: Persistent Stored Modules (SQL/PSM)
- Part 7: Temporal (SQL/Temporal)
- Part 9: Management of External Data (SQL/MED)
- Part 10: Object Language Bindings (SQL/OLB)
- Part 11: Information and Definition Schema (SQL/Schemata)

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

To be supplied if required.

Annexes A, B, C, and D of this part of

ANSI

 ANSI X3.135

ISO

 ISO/IEC 9075
are for information only.

Introduction

The organization of this Part of this International Standard is as follows:

- 1) Clause 1, “Scope”, specifies the scope of this part of this International Standard.
- 2) Clause 2, “Normative references”, identifies additional standards that, through reference in this part of this International Standard, constitute provisions of this part of this International Standard.
- 3) Clause 3, “Definitions, notations, and conventions”, defines the notations and conventions used in this part of this International Standard.
- 4) Clause 4, “Concepts”, presents concepts related to this Parts of this International standard related to the support of time.
- 5) Clause 5, “Lexical elements”, defines a number of lexical elements used in the definition of temporal support.
- 6) Clause 6, “Scalar expressions”, defines a number of scalar expressions used in the definition of temporal support.
- 7) Clause 7, “Query expressions”, defines the elements of the language that produce rows and tables of data, enhances for temporal support.
- 8) Clause 8, “Predicates”, defines a number of predicates used in the manipulation of temporal support.
- 9) Clause 10, “Additional common rules”, specifies the rules for assignments that retrieve data from or store data into SQL-data, and formation fules for set operations.
- 10) Clause 11, “Schema definition and manipulation”, defines facilities for creating and managing a schema.
- 11) Clause 13, “Dynamic SQL”, defines temporal support in the SQL dynamic statements.
- 12) Clause 17, “Definition Schema”, defines base tables on which the viewed tables containing schema information depend.
- 13) Clause 18, “Status codes”, defines SQLSTATE values related to temporal support.
- 14) Clause 19, “Conformance”, defines the criteria for conformance to this Part of this International Standard.
- 15) Annex A, “Implementation-defined elements”, is an informative Annex. It lists those features for which this part of

ANSI	ANSI X3.135
ISO	ISO/IEC 9075

 states that the syntax, the meaning, the returned results, the effect on SQL-data and/or schemas, or any other behavior is partly or wholly implementation-defined.

- 16) Annex B, “Implementation-dependent elements”, is an informative Annex. It lists those features for which the body of this part of

ANSI	ANSI X3.135
------	-------------

ISO	ISO/IEC 9075
-----	--------------

states that the syntax, the meaning, the returned results, the effect on SQL-data and/or schemas, or any other behavior is partly or wholly implementation-dependent.
- 17) Annex C, “Deprecated features”, is an informative Annex. It lists features that the responsible Technical Committee intends will not appear in a future revised version of this International Standard.
- 18) Annex D, “Incompatibilities with ISO/IEC 9075:1999”, is an informative Annex. It lists the incompatibilities between this version of this International Standard and ISO/IEC 9075:1992.

In the text of this International Standard, Clauses begin a new odd-numbered page. Any resulting blank space is not significant.

Information technology — Database languages — SQL — Part 7: Temporal (SQL/Temporal)

1 Scope

This Part of International Standard ISO/IEC 9075 defines the facilities by which a conforming SQL-implementation can support temporal data. The database language for temporal support includes:

- the specification of a data type constructor named PERIOD; and
- the specification of scalar expressions and predicates used to manipulate values of period data types.

NOTE 1 – The framework for this International Standard is described by the Reference Model of Data Management (ISO/IEC 10032:1993).

SC32 N00651 = WG3:YYJ-007 = H2-2001-144

2 Normative references

The following normative documents contain provisions that, through reference in this text, constitute provisions of this International Standard. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this International Standard are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

ISO/IEC 1539:1991, *Information technology — Programming languages — Fortran*.

ISO 1989:1985, *Programming languages — COBOL*.

ISO 6160:1979, *Programming languages — PL/I*.

ISO/IEC 7185:1990, *Information technology — Programming languages — Pascal*.

ISO 8652:1987, *Programming languages — Ada*.

ISO/IEC FCD 9075-1:200x, *Information technology — Database languages — SQL — Part 1: Framework (SQL/Framework)*.

ISO/IEC FCD 9075-2:200x, *Information technology — Database languages — SQL — Part 2: Foundation (SQL/Foundation)*.

ISO/IEC FCD 9075-3:200x, *Information technology — Database languages — SQL — Part 3: Call-Level Interface (SQL/CLI)*.

ISO/IEC FCD 9075-4:200x, *Information technology — Database languages — SQL — Part 4: Persistent Stored Modules (SQL/PSM)*.

ISO/IEC FCD 9075-9:200x, *Information technology — Database languages — SQL — Part 9: Management of External Data (SQL/MED)*.

ISO/IEC FCD 9075-10:200x, *Information technology — Database languages — SQL — Part 10: Object Language Bindings (SQL/OLB)*.

ISO/IEC FCD 9075-11:200x, *Information technology — Database languages — SQL — Part 11: Information and Definition Schemas (SQL/Schemata)*.

ISO/IEC FCD 9075-13:200x, *Information technology — Database languages — SQL — Part 13: Java Routines and Types (SQL/JRT)*.

ISO/IEC FCD 9075-14:200x, *Information technology — Database languages — SQL — Part 14: XML-Related Specifications (SQL/XML)*.

ISO/IEC 9899:1999, *Information technology — Programming languages — C*.

SC32 N00651 = WG3:YYJ-007 = H2-2001-144

ISO/IEC 10206:1991, *Information technology — Programming languages — Extended Pascal.*

ISO/IEC 11756:1992, *Information technology—Programming languages—MUMPS.*

3 Definitions, notations, and conventions

3.1 Definitions

For the purposes of this International Standard, the following definitions apply.

All definitions in Parts 1 and 2 of this International Standard apply to this Part or this International Standard.

This Part of this International Standard defines the following terms:

- a) **beginning bound (of a period)**: the least value in a period.
- b) **element type**: the data type of the elements of a value of either a collection data type or a period data type.

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

- c) **ending bound (of a period)**: the least value of the element type that is greater than the last element of a period.
- d) **granule**: a period of the minimum duration representable at a given precision; *i.e.*, containing a single element. For PERIOD (TIMESTAMP (P)), the duration of a granule is 10^{-P} seconds. A granule may be denoted by a single value of a data type, or by a period whose last element is equal to its beginning bound.
- e) **last element (of a period)**: The element that is greater than every other element.
- f) **period**: a convex set of values P of some well ordered (datetime) data type (known as the element type). That is to say: if two values, $V1$ and $V2$ are in P , and there is a value $V3$ such that $V1 \leq V3 \leq V2$, then $V3$ is in P .

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

- g) **ending bound (of a period)**: the greatest value in a period.
 - h) **period**: a convex set of values P of some well ordered data type (known as the element type). That is to say: if two values, $V1$ and $V2$ are in P , and there is a value $V3$ such that $V1 \leq V3 \leq V2$, then $V3$ is in P .
 - i) **value-equivalent** (with respect to . . .): Let $R1$ and $R2$ be rows of the same row type having fields F_i , 1 (one) $\leq i \leq n$ for some n . If, disregarding the fields F_i (F_j . . .), the rows formed from the remaining fields of $R1$ and $R2$, taken in the order of their corresponding columns, are duplicates, then $R1$ and $R2$ are said to be value-equivalent rows with respect to F_i (F_j . . .).
-

3.2 Notations

3.2 Notations

All notations in Part 1 and Part 2 of this International Standard apply to this Part. The syntax notation used in this Part of this International Standard is an extended version of BNF (“Backus Normal Form” or “Backus Naur Form”). This version of BNF is fully described in Part 2 of this International Standard.

3.3 Conventions

Except as otherwise specified in this Part of this International Standard, the conventions used in this Part of this International Standard, are identical to those described in Subclause 3.3, “Conventions”, of Parts 1 and 2 of this International Standard.

The contents of this Part of this International Standard depend wholly on Part 2 of this International Standard. For example, the Syntax found in the Format portions of this Part of this International Standard often uses symbols that are defined in Part 2 of this International Standard.

3.3.1 Relationships to other parts of ISO/IEC 9075

This part of ISO/IEC 9075 depends on part 2 of ISO/IEC 9075 and its Technical Corrigenda. This part of ISO/IEC 9075 is to be used as though it were merged with the text of part 2 of ISO/IEC 9075. This Subclause describes the conventions used to specify the merger. The merger described also accounts for the Technical Corrigenda that have been published to correct part 2 of ISO/IEC 9075. This accommodation is typically indicated by the presence of a phrase like “in the Technical Corrigenda” or “in the TC”.

3.3.1.1 New and modified Clauses, Subclauses, and Annexes

Clauses, Subclauses, and Annexes (other than Clause 1, “Scope”, and Clause 2, “Normative references”) in this part of ISO/IEC 9075 that have names identical to Clauses, Subclauses, or Annexes in part 2 of ISO/IEC 9075 supplement the Clause, Subclause, or Annex, respectively, in part 2 of ISO/IEC 9075, typically by replacing paragraphs, Format items, or Rules or by providing new paragraphs, Format items, or Rules. However, Clauses, Subclauses, and Annexes in this part of ISO/IEC 9075 that have names identical to Clauses, Subclauses, and Annexes in part 2 of ISO/IEC 9075 do not necessarily have the same *number* or *letter* as the corresponding Clause, Subclause, or Annex in part 2 of ISO/IEC 9075. Any differences in Clause or Subclause number or in Annex letter is not significant. Table 1, “Clause, Subclause, and Table relationships”, identifies the relationships between Clauses, Subclauses, and Annexes in this part of ISO/IEC 9075 and the corresponding Clauses, Subclauses, and Annexes in other parts of ISO/IEC 9075.

Clauses, Subclauses, and Annexes in this part of ISO/IEC 9075 that have names that are not identical to Clauses, Subclauses, or Annexes in part 2 of ISO/IEC 9075 provide language specification particular to this part of ISO/IEC 9075. Subclauses that are subsidiary to (“contained in”) Clauses or Subclauses identified as new are inherently new and are not further identified.

The Clauses, Subclauses, and Annexes in this part of ISO/IEC 9075 appear in the order in which they are intended to appear in the merged document. Absent other explicit instructions regarding its placement, any new Clause, Subclause, or Annex is to be positioned as follows: Locate the prior Clause, Subclause, or Annex in this part of ISO/IEC 9075 whose name is identical to the name of a corresponding Clause, Subclause, or Annex that appears in another part of ISO/IEC 9075. The new

Clause, Subclause, or Annex shall immediately follow that Clause, Subclause, or Annex. If there are multiple new Clauses, Subclauses, or Annexes with no intervening Clause, Subclause, or Annex that modifies an existing Clause, Subclause, or Annex, then those new Clauses, Subclauses, or Annexes appear in order, following the prior Clause, Subclause, or Annex whose name was matched.

3.3.1.2 New and modified Format items

In modified Subclauses, Format items that define a BNF nonterminal symbol (that is, the BNF nonterminal symbol appears on the left-hand side of the ::= mark) sometimes modify a Format item whose definition appears in part 2 of ISO/IEC 9075, sometimes replace a Format item whose definition appears in part 2 of ISO/IEC 9075, and sometimes define a new Format item that does not have a definition at all in part 2 of ISO/IEC 9075. Those Format items in this part of ISO/IEC 9075 that modify a Format item whose definition appears in part 2 of ISO/IEC 9075, are identified by the existence of a “Format comment” such as:

```
<modified item> ::=  
    !! All alternatives from ISO/IEC 9075-2  
    | <new alternative>
```

By contrast, Format items that completely replace Format items in part 2 of ISO/IEC 9075 have BNF nonterminal symbols identical to BNF nonterminal symbols of Format items in part 2 of ISO/IEC 9075, but do not state that they include any alternatives from part 2 of ISO/IEC 9075.

New Format items that have no correspondence to any Format item in part 2 of ISO/IEC 9075 are not specially identified in this part of ISO/IEC 9075.

In new Subclauses, all Format items are also new and require no specific marking.

3.3.1.3 New and modified paragraphs and rules

In modified Subclauses, each paragraph or Rule is marked to indicate whether it is a modification of a paragraph or Rule in part 2 of ISO/IEC 9075, or is a new paragraph or Rule in this part of ISO/IEC 9075.

Modifications of paragraphs or Rules in part 2 of ISO/IEC 9075, are identified by the inclusion of an indication such as `Replace the 5th paragraph`, meaning that the fifth paragraph of the corresponding Subclause in part 2 of ISO/IEC 9075, is to be replaced by the paragraph to which the indication is attached, or `Replace SR6)b)ii)`, meaning that Syntax Rule 6)b)ii) of the corresponding Subclause in part 2 of ISO/IEC 9075, is to be replaced by the rule to which the indication is attached. In some cases, an indication such as `Augments SR3)` is used. When this appears, it means that the referenced Rule is extended or enhanced by the Rule to which the indication is attached. In most instances, the augmentation is the addition of a new alternative meant to support new syntax.

New paragraphs or Rules in this part of ISO/IEC 9075 are indicated by the inclusion of an indication such as `Insert before 2nd paragraph`, meaning that the paragraph to which the indication is attached is to be read as though it were inserted preceding the second paragraph of the corresponding Subclause in part 2 of ISO/IEC 9075. `Insert before GR4)` should be interpreted to mean that the rule to which the indication is attached is to be read as though it were inserted preceding General Rule 4) of the corresponding Subclause in part 2 of ISO/IEC 9075. When an indication does not indicate a specific insertion point, such as `Insert this paragraph` or `Insert this GR`, then it may be read as implicitly specifying that the new text is to be appended at the end of the appropriate section (the General Rules, for example) of the corresponding Subclause in part 2 of ISO/IEC 9075.

3.3 Conventions

In such instructions, “SR” is used to mean “Syntax Rule”, “AR” is used to mean “Access Rule”, and “GR” is used to mean “General Rule”. “Desc.” is used to mean “Description” and “Func.” is used to mean “Function”.

All paragraphs, Format items, and Rules in new Clauses or Subclauses are also new and therefore do not require further identification.

3.3.1.4 New and modified tables

Similarly, tables in this part of ISO/IEC 9075 that have names that are identical to tables in part 2 of ISO/IEC 9075 supplement the table in part 2 of ISO/IEC 9075, typically by adding or replacing one or more table entries. Tables in this part of ISO/IEC 9075 that have names that are not identical to the names of tables in part 2 of ISO/IEC 9075 are new tables specified by this part of ISO/IEC 9075. Table 1, “Clause, Subclause, and Table relationships”, identifies the relationships between tables in this part of ISO/IEC 9075 and the corresponding tables in other parts of ISO/IEC 9075.

The rows in modified tables are generally new rows to be effectively inserted into the corresponding table in part 2 of ISO/IEC 9075, though in rare cases rows already in tables in part 2 of ISO/IEC 9075 are effectively replaced by rows in the table in this part of ISO/IEC 9075. It is always obvious when such an effective replacement is required, based on equality of values in the first column of the table.

3.3.1.5 Clause, Subclause, and Table relationships

Table 1—Clause, Subclause, and Table relationships

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part	Part containing correspondence
Clause 1, “Scope”	Clause 1, "Scope"	ISO/IEC 9075-2
Clause 2, “Normative references”	Clause 2, "Normative references"	ISO/IEC 9075-2
Clause 3, “Definitions, notations, and conventions”	Clause 3, "Definitions, notations, and conventions"	ISO/IEC 9075-2
Subclause 3.1, “Definitions”	Subclause 3.1, "Definitions"	ISO/IEC 9075-2
Subclause 3.2, “Notations”	Subclause 3.2, "Notation"	ISO/IEC 9075-2
Subclause 3.3, “Conventions”	Subclause 3.3, "Conventions"	ISO/IEC 9075-2
Subclause 3.3.1, “Relationships to other parts of ISO/IEC 9075”	<i>(none)</i>	<i>(none)</i>
Subclause 3.3.1.1, “New and modified Clauses, Subclauses, and Annexes”	<i>(none)</i>	<i>(none)</i>
Subclause 3.3.1.2, “New and modified Format items”	<i>(none)</i>	<i>(none)</i>
Subclause 3.3.1.3, “New and modified paragraphs and rules”	<i>(none)</i>	<i>(none)</i>

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

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part	Part containing correspondence
Subclause 3.3.1.4, “New and modified tables”	<i>(none)</i>	<i>(none)</i>
Subclause 3.3.1.5, “Clause, Subclause, and Table relationships”	<i>(none)</i>	<i>(none)</i>
Subclause 3.4, “Object identifier for Database Language SQL”	Subclause 6.3, "Object identifier for Database Language SQL"	ISO/IEC 9075-1
Clause 4, “Concepts”	Clause 4, "Concepts"	ISO/IEC 9075-2
Subclause 4.1, “Introduction to SQL/Temporal”	<i>none</i>	<i>none</i>
Subclause 4.2, “Period data type”	<i>none</i>	<i>none</i>
Subclause 4.2.1, “Operations involving periods”	<i>none</i>	<i>none</i>

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

Subclause 4.2.1.1, “Functions that operate on periods and return periods”	<i>none</i>	<i>none</i>
Subclause 4.2.1.2, “Functions that operate on periods and return values of element type”	<i>none</i>	<i>none</i>
Subclause 4.2.1.3, “Other operations involving period elements”	<i>none</i>	<i>none</i>
Subclause 4.2.1.4, “Other operations involving periods”	<i>none</i>	<i>none</i>

Subclause 4.2.2, “Period type conversions and mixing of data types”	Subclause 4.12, "Type conversions and mixing of data types"	ISO/IEC 9075-2
Subclause 4.2.3, “Period predicates”	<i>none</i>	<i>none</i>

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

Subclause 4.3, “Tables”	Subclause 4.16, "Tables"	ISO/IEC 9075-2
Subclause 4.3.1, “Operations involving tables”	Subclause 4.16.3, "Operations involving tables"	ISO/IEC 9075-2

3.3 Conventions

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

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part	Part containing correspondence
Subclause 4.3.1.1, "Operations involving tables with period columns"	<i>none</i>	<i>none</i>
Subclause 4.4, "Standard programming languages"	Subclause 4.33, "Standard programming languages"	ISO/IEC 9075-2
Subclause 4.5, "Miscellaneous characteristics"	Subclause 4.4, "Miscellaneous characteristics"	ISO/IEC 9075-3
Subclause 4.5.1, "Statement attributes"	Subclause 4.4.6, "Statement attributes"	ISO/IEC 9075-3
<hr/>		
Clause 5, "Lexical elements"	Clause 5, "Lexical elements"	ISO/IEC 9075-2
Subclause 5.1, "<token> and <separator>"	Subclause 5.2, "<token> and <separator>"	ISO/IEC 9075-2
Subclause 5.2, "<literal>"	Subclause 5.3, "<literal>"	ISO/IEC 9075-2
Clause 6, "Scalar expressions"	Clause 6, "Scalar expressions"	ISO/IEC 9075-2
Subclause 6.1, "<data type>"	Subclause 6.1, "<data type>"	ISO/IEC 9075-2
Subclause 6.2, "<set function specification>"	Subclause 6.16, "<set function specification>"	ISO/IEC 9075-2
Subclause 6.3, "<cast specification>"	Subclause 6.23, "<cast specification>"	ISO/IEC 9075-2
Subclause 6.4, "<value expression>"	Subclause 6.24, "<value expression>"	ISO/IEC 9075-2
<hr/>		
ANSI Only—caused by ISO changes not yet considered by ANSI		
<hr/>		
Subclause 6.5, "<period value expression>"	<i>none</i>	<i>none</i>
Subclause 6.6, "<set or multiset value expression>"	<i>none</i>	<i>none</i>
<hr/>		
Subclause 6.7, "<period set value function>"	<i>none</i>	<i>none</i>
Subclause 6.8, "<period value constructor>"	<i>none</i>	<i>none</i>
Clause 7, "Query expressions"	Clause 7, "Query expressions"	ISO/IEC 9075-2

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

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part	Part containing correspondence
Subclause 7.1, “<normalized query specification>”	<i>none</i>	<i>none</i>
Subclause 7.2, “<query expression>”	Subclause 7.13, “<query expression>”	ISO/IEC 9075-2
Subclause 7.3, “<expanding clause>”	<i>none</i>	<i>none</i>
Clause 8, “Predicates”	Clause 8, “Predicates”	ISO/IEC 9075-2
Subclause 8.1, “<predicate>”	Subclause 8.1, “<predicate>”	ISO/IEC 9075-2
Subclause 8.2, “<comparison predicate>”	Subclause 8.2, “<comparison predicate>”	ISO/IEC 9075-2

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

Subclause 8.3, “<overlaps predicate>”	Subclause 8.12, “<overlaps predicate>”	ISO/IEC 9075-2
---------------------------------------	--	----------------

Subclause 8.4, “<period predicate>”	<i>none</i>	<i>none</i>
-------------------------------------	-------------	-------------

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

Clause 9, “Expansion and compression”	<i>none</i>	<i>none</i>
Subclause 9.1, “Expansion of a table”	<i>none</i>	<i>none</i>
Subclause 9.2, “Compression of a table”	<i>none</i>	<i>none</i>

Clause 10, “Additional common rules”	Clause 9, “Additional common rules”	ISO/IEC 9075-2
Subclause 10.1, “Retrieval assignment”	Subclause 9.1, “Retrieval assignment”	ISO/IEC 9075-2
Subclause 10.2, “Store assignment”	Subclause 9.2, “Store assignment”	ISO/IEC 9075-2
Subclause 10.3, “Set operation result data types”	Subclause 9.3, “Data types of results of aggregations”	ISO/IEC 9075-2

3.3 Conventions

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

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part	Part containing correspondence
Clause 11, "Schema definition and manipulation"	Clause 11, "Schema definition and manipulation"	ISO/IEC 9075-2
Subclause 11.1, "<default clause>"	Subclause 11.5, "<default clause>"	ISO/IEC 9075-2

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

Clause 12, "SQL-client modules"	Clause 13, "SQL-client modules"	ISO/IEC 9075-2
Subclause 12.1, "Data type correspondences"	Subclause 13.6, "Data type correspondences"	ISO/IEC 9075-2

Clause 13, "Dynamic SQL"	Clause 19, "Dynamic SQL"	ISO/IEC 9075-2
Subclause 13.1, "Description of SQL item descriptor areas"	Subclause 19.1, "Description of SQL descriptor areas"	ISO/IEC 9075-2

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

Subclause 13.2, "<get descriptor statement>"	Subclause 19.4, "<get descriptor statement>"	ISO/IEC 9075-2
--	--	----------------

Subclause 13.3, "<set descriptor statement>"	Subclause 19.5, "<set descriptor statement>"	ISO/IEC 9075-2
--	--	----------------

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

Subclause 13.4, "<prepare statement>"	Subclause 19.6, "<prepare statement>"	ISO/IEC 9075-2
---------------------------------------	---------------------------------------	----------------

Subclause 13.5, "<describe statement>"	Subclause 19.8, "<describe statement>"	ISO/IEC 9075-2
--	--	----------------

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

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

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part	Part containing correspondence
Clause 14, “Call-Level Interface specifications”	Clause 5, "Call-Level Interface specifications"	ISO/IEC 9075-3
Subclause 14.1, “Implicit EXECUTE USING and OPEN USING clauses”	Subclause 5.6, "Implicit EXECUTE USING and OPEN USING clauses"	ISO/IEC 9075-3
Subclause 14.2, “Description of CLI item descriptor areas”	Subclause 5.13, "Description of CLI item descriptor areas"	ISO/IEC 9075-3
Subclause 14.3, “SQL/CLI data type correspondences”	Subclause 5.15, "SQL/CLI data type correspondences"	ISO/IEC 9075-3
Clause 15, “SQL/CLI routines”	Clause 6, "SQL/CLI routines"	ISO/IEC 9075-3
Subclause 15.1, “GetData”	Subclause 6.30, "GetData"	ISO/IEC 9075-3
Subclause 15.2, “GetParamData”	Subclause 6.40, "GetParamData"	ISO/IEC 9075-3
Subclause 15.3, “SetDescField”	Subclause 6.56, "SetDescField"	ISO/IEC 9075-3
Clause 16, “Information Schema”	Clause 5, "Information Schema"	ISO/IEC 9075-11
Subclause 16.1, “ELEMENT_TYPES view”	Subclause 5.28, "ELEMENT_TYPES view"	ISO/IEC 9075-11
<hr/>		
Clause 17, “Definition Schema”	Clause 6, "Definition Schema"	ISO/IEC 9075-11
Subclause 17.1, “DATA_TYPE_DESCRIPTOR base table”	Subclause 6.16, "DATA_TYPE_DESCRIPTOR base table"	ISO/IEC 9075-11
<hr/>		
ISO Only—caused by ANSI changes not yet considered by ISO		
<hr/>		
Subclause 17.2, “ELEMENT_TYPES base table”	Subclause 6.21, "ELEMENT_TYPES base table"	ISO/IEC 9075-11
<hr/>		
Clause 18, “Status codes”	Clause 23, "Status codes"	ISO/IEC 9075-2
Subclause 18.1, “SQLSTATE”	Subclause 23.1, "SQLSTATE"	ISO/IEC 9075-2
Clause 19, “Conformance”	Clause 24, "Conformance"	ISO/IEC 9075-2
Annex A, “Implementation-defined elements”	Appendix B, "Implementation-defined elements"	ISO/IEC 9075-2
Annex B, “Implementation-dependent elements”	Appendix C, "Implementation-dependent elements"	ISO/IEC 9075-2
Annex C, “Deprecated features”	Appendix D, "Deprecated features"	ISO/IEC 9075-2

3.3 Conventions

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

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part	Part containing correspondence
Annex D, "Incompatibilities with ISO/IEC 9075:1999"	Appendix E, "Incompatibilities with ISO/IEC 9075-2:1999 and ISO/IEC 9075-4:1999"	ISO/IEC 9075-2
Table 9, "Codes used for SQL data types in Dynamic SQL"	Table 27, "Codes used for SQL data types in Dynamic SQL"	ISO/IEC 9075-2
Table 10, "Codes associated with datetime and period data types in Dynamic SQL"	Table 28, "Codes associated with datetime data types in Dynamic SQL"	ISO/IEC 9075-2
Table 19, "SQLSTATE class and subclass values"	Table 34, "SQLSTATE class and subclass values"	ISO/IEC 9075-2

3.4 Object identifier for Database Language SQL

NOTE 2 – It is possible that the object identifier for SQL may have to be adjusted to account for the new structure of SQL3.

4 Concepts

4.1 Introduction to SQL/Temporal

To Be Supplied.

4.2 Period data type

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

A period data type is described by a period data type descriptor. A period data type has an element type that may be any datetime data type. The period data type descriptor for period data type *PT* includes:

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

A period type is described by a period data type descriptor. A period type has an element type that may be any datetime data type or exact numeric type with scale zero. The period data type descriptor for period data type *PT* includes:

— An indication of the element type of *PT*.

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

A value of that data type obtained by subtracting two values of the element type of a period may be added to or subtracted from a period, the result being a period of the same type. No other arithmetic operations on periods are permitted.

A period is a representation of a set of contiguous granules of a specified precision.

For a period with data type `PERIOD(TIMESTAMP(P))`, a granule is one particular 10^{-P} second. For example, `PERIOD(0)` is a set of contiguous seconds, with each granule being a particular second.

The first and last granules of a period, termed the beginning and ending delimiting timestamps, respectively, of the period, are `TIMESTAMPS` of the same precision as the period. The delimiting timestamps can be extracted using `BEGIN` and `END`. If `WITH TIME ZONE` is specified for the period, then the `TIMESTAMPS` returned by `BEGIN` and `END` include the timezone field.

4.2 Period data type

The functions BEGIN, END, and LAST return respectively the value of the beginning bound, the value of the ending bound, and the value that denotes the last granule of the period. Thus, if the data type of $P1$ is PERIOD(TIMESTAMP(0)), then LAST($P1$) is the value of TIMESTAMP that denotes the beginning of the last second of $P1$, while END($P1$) is one second later. END($P1$) is actually the same as BEGIN($P2$), where $P2$ is some period that is said to *meet* $P1$; that is, it follows it without overlapping, and without any intervening gap.

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

If the data type of the result of subtracting any two values of the element type of a period P is DT , then a value of DT may be added to or subtracted from P , the result being a period of the same type. No other arithmetic operations on periods are permitted.

The first and last elements of a period, termed the beginning bound and ending bound, respectively, of the period, are values of the element type of the period.

4.2.1 Operations involving periods

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

Normalization is an operation that maps a set of periods to exactly one set of periods that is equivalent, in the sense that every granule that is a member of some element of the operand is a member of exactly one element in the result, and minimal, in the sense that no two elements $P1$ and $P2$ of the result are such that $P1$ meets $P2$. (The term “meet” is defined in Subclause 4.2.3, “Period predicates”.) The NORMALIZE function defined in this International Standard takes as its argument a set or multiset of periods; the result is the normalized set of periods.

Because the NORMALIZE function operates only on sets (or multisets) rather than tables, a <normalize clause> is also defined, in <table expression>, so that tables with columns of data type PERIOD can be normalized with respect to one or more such columns. Let T be a table containing a period-valued column P . If, disregarding the values in P , the rows $R1$ and $R2$ in T are duplicates, then $R1$ and $R2$ are said to be *value-equivalent rows* with respect to P . When T is normalized on P , the result R is a table in which:

- R and T have the same row type.
- With respect to P , every row in T is value-equivalent to some row in R , and every row in R is value-equivalent to some row in T .
- No two distinct but value-equivalent rows in R have P values that intersect or meet each other. (The terms “intersect” and “meet” are defined in Subclause 4.2.3, “Period predicates”.)

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

4.2.1.1 Functions that operate on periods and return periods

There are three dyadic functions that take operands $P1$ and $P2$ of period type and return a result PR of period type. All require that $P1$ intersects $P2$.

NOTE 3 – “intersects” is defined in Subclause 4.2.3, “Period predicates”.

— P_UNION returns the period PR such that, if and only if $P1$ contains a period P or $P2$ contains P , then PR contains P .

NOTE 4 – “contains” is defined in Subclause 4.2.3, “Period predicates”.

— P_EXCEPT returns the period PR such that, if and only if $P1$ contains a period P and $P2$ does not contain P , then PR contains P .

— $P_INTERSECT$ returns the period PR such that, if and only if $P1$ contains a period P and $P2$ contains P , then PR contains P .

4.2.1.2 Functions that operate on periods and return values of element type

<period bound function> is a pair of monadic functions, $FIRST$ and $LAST$, that return, respectively, the values of the first and last elements of their period argument. The declared type of the result is the element type of the period.

4.2.1.3 Other operations involving period elements

<proximity function> is a pair of monadic functions, $NEXT$ and $PRIOR$, whose argument is of a period element type. $NEXT$ returns the smallest value of the argument type that is greater than the argument value. $PRIOR$ returns the largest value of the argument type that is less than the argument value.

4.2.1.4 Other operations involving periods

<period duration function> is a monadic function, $DURATION$, that returns the duration of its period argument. The declared type of the result is the data type that would result from the subtraction of two values of the element type.

4.2.2 Period type conversions and mixing of data types

Values of type period are mutually assignable if and only if their respective element types are mutually assignable, and are mutually comparable if and only if their respective element types are mutually comparable.

Two periods are ordered on their beginning bounds, or, if those are equal, their ending bounds.

4.2 Period data type

4.2.3 Period predicates

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

Period $P1$ *precedes* period $P2$ if $LAST(P1)$ is less than $BEGIN(P2)$. Period $P1$ *contains* period $P2$ if the $BEGIN(P1)$ is less than or equal to $BEGIN(P2)$ and $LAST(P1)$ is greater than or equal to $LAST(P2)$. Period $P1$ *meets* period $P2$ if $END(P1)$ is equal to $BEGIN(P2)$, or $END(P2)$ is equal to $BEGIN(P1)$. Period $P1$ *overlaps* period $P2$ if $BEGIN(P1)$ is less than or equal to $LAST(P2)$ and $BEGIN(P2)$ is less than or equal to $BEGIN(P1)$, or vice versa.

NOTE 5 – “overlaps” is defined in ISO/IEC 9075-2, and this definition is compatible.

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

Period $P1$ *precedes* period $P2$ if $LAST(P1)$ is less than $FIRST(P2)$. Period $P1$ *contains* period $P2$ if $FIRST(P1)$ is less than or equal to $FIRST(P2)$ and $LAST(P1)$ is greater than or equal to $LAST(P2)$. Period $P1$ *meets* period $P2$ if $NEXT(LAST(P1))$ is equal to $FIRST(P2)$, or $NEXT(LAST(P2))$ is equal to $FIRST(P1)$. Period $P1$ *intersects* period $P2$ if $FIRST(P1)$ is less than or equal to $LAST(P2)$ and $FIRST(P2)$ is less than or equal to $FIRST(P1)$, or vice versa.

4.3 Tables

4.3.1 Operations involving tables

4.3.1.1 Operations involving tables with period columns

<expanding clause> is an option of <query expression> that can be specified in the case that both operands of one of the set operators UNION, EXCEPT or INTERSECT are tables that have one or more period columns referenced by the same <column references>s. The effect of <expanding clause> is to expand both operands before applying the set operator and, after applying it, to compress its result.

NOTE 6 – The primitive operations expansion and compression are described later in this subclause.

A <normalized query specification> contains a <query specification> followed by a <normalize clause>. The result of the <normalized query specification> is the result of the <query specification> first expanded and then compressed on those columns specified in the <normalizing column reference list>.

Two primitive operations, expansion and compression, are specified for the sole purpose of simplifying the definitions of <expanding clause> and <normalized query specification>.

Expansion and compression each require two operands: a table $T1$, one or more of whose columns is of a period type, and a list of references to one or more such columns. In each case, the result is a table $T2$ having the same row type.

The result of expanding *T1* on one column *C* is *T2*, such that:

- Let *R* be some row of *T1* and let the value for *C* in *R* be *P*. The expanded result set of *R* consists of rows, every one of which is value equivalent to *R* with respect to *C*, no two of which are duplicates, and the value, *PR* for *C* takes every possible value such that *P* contains *PR* and $\text{FIRST}(PR) = \text{LAST}(PR)$. (The term “contains” is defined in Subclause 4.2.3, “Period predicates”.)
- *T2* comprises the union of the expanded result set of every row of *T1*.

The result of compressing *T1* on one column *C* is *T2*, such that:

- With respect to *C*, every row in *T2* is value-equivalent to some row in *T1*, and every row in *T1* is value-equivalent to some row in *T2*.
- No two distinct rows in *T2* that are value-equivalent with respect to *C* have values for *C* that intersect or meet each other. (The terms “intersect” and “meet” are defined in Subclause 4.2.3, “Period predicates”.)

Expansion or compression on more than one column is effectively performed on each column referenced in the list in turn.

4.4 Standard programming languages

Insert this paragraph There is no data type in any standard programming language that corresponds to a period type.

NOTE 7 – For the purposes of fetching a period value into a host language application, it is sufficient to fetch the bounds of a period value into a pair of host variables whose data type corresponds to the element type of the period; conversely, a period value can be inserted or updated by using a period value constructor. Where the element type is a datetime type, it requires to be dealt with accordingly (see Subclause 4.33, “Standard programming languages”, in ISO/IEC 9075-2). It is expected that future evolution of programming language standards will support data types corresponding to this SQL data type; this standard will then be amended to reflect the availability of those corresponding data types.

4.5 Miscellaneous characteristics

4.5.1 Statement attributes

Replace 11th paragraph The NEST DESCRIPTOR attribute determines whether nested descriptor items are permitted in a CLI descriptor. Nested descriptor items are used to describe ROW, ARRAY, and PERIOD data types. The attribute is set to FALSE when the statement is allocated.

SC32 N00651 = WG3:YYJ-007 = H2-2001-144

5 Lexical elements

5.1 <token> and <separator>

Function

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

Format

```
<delimiter token> ::=
    !! All alternatives from ISO/IEC 9075-2
    | <period string>
```

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

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

!! All alternatives from ANSI X3.135.2

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

!! All alternatives from ISO/IEC 9075-2

```
<reserved word> ::=
```

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

!! All alternatives from ANSI X3.135.2

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

!! All alternatives from ISO/IEC 9075-2

| CONTAINS

| EXPAND | EXPANDING

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

| INTERSECTS

| MEETS

| NORMALIZE

SC32 N00651 = WG3:YYJ-007 = H2-2001-144

5.1 <token> and <separator>

| PERIOD | PRECEDES

| SUCCEEDS

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

5.2 <literal>

Function

Specify a non-null value.

Format

```
<general literal> ::=  
    !! All alternatives from ISO/IEC 9075-2  
    | <period literal>  
  
<period literal> ::= PERIOD <period string>  
  
<period string> ::=  
    <quote> <beginning period delimiter> <beginning value>  
    <period bound separator>  
    <ending value> [ <time zone interval> ]  
    <ending period delimiter> <quote>  
  
<beginning period delimiter> ::=  
    <left bracket>  
    | <left paren>  
  
<ending period delimiter> ::=  
    <right bracket>  
    | <right paren>  
  
<period bound separator> ::=  
    <space> <minus sign> <space>  
    | [ <space> ] <comma> [ <space> ]  
  
<beginning value> ::=  
    <date value> <space> <time value>  
    | <date value>  
    | <time value>  
  
<ending value> ::=  
    <date value> <space> <time value>  
    | <date value>  
    | <time value>
```

Syntax Rules

- 1) Insert this SR Let *BPD* be <beginning period delimiter>, let *BV* be <beginning value>, let *EV* be <ending value>, and let *EPD* be <ending period delimiter>. If <time zone interval> is specified, then let *TZI* be <time zone interval>; otherwise, let *TZI* be the empty string.
- 2) Insert this SR Case:
 - a) If *BV* contains both <date value> and <time value>, then *EV* shall also contain both <date value> and <time value>. Let *PET* be `TIMESTAMP`.
 - b) If *BV* contains <date value>, then *EV* shall contain <date value> and shall not contain <time value>. Let *PET* be `DATE`.

SC32 N00651 = WG3:YYJ-007 = H2-2001-144

5.2 <literal>

- c) Otherwise, *BV* shall contain <time value> and *EV* shall contain <time value> and shall not contain <date value>. Let *PET* be TIME.

- 3) Insert this SR The <period literal> is equivalent to:

`PERIOD BPD PET 'BV TZI' , PET 'ET TZI' EPD`

Access Rules:

No additional Access Rules.

General Rules

No additional General Rules.

6 Scalar expressions

6.1 <data type>

Function

Specify a data type.

Format

```
<predefined type> ::=
    !! All alternatives from ISO/IEC 9075-2
    | <period type>

<period type> ::=
    PERIOD <left paren> <data type> <right paren>
```

Syntax Rules

- 1) Insert this SR If PERIOD is specified, then the <data type> immediately contained in <period type> shall be a datetime data type or shall be exact numeric with a scale of 0 (zero).

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

6.2 <set function specification>

Function

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

Format

No additional Format items.

Syntax Rules

- 1)

Replaces SR<REFERENCE>(fnd_func_dt_SR\FULL))
--

DT shall not be a character string, bit string, an enumerated type, datetime, or a period type.

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

The preceding Rule references a Rule that used to be in Subclause 6.16, "<set function specification>", but that (because of the changes made by SQL/OLAP, both as an amendment to Foundation:1999 and by means of the merger with Foundation:200n in WG3:RTM-054R2/X3H2-99-310R3) has now moved to a new Subclause: Subclause 10.10, "<aggregate function>", where it is now SR7)g)i)). Thus, we see yet again the effects of the law of unintended consequences—SQL/Temporal will have to be restructured to take into account the SQL/OLAP restructuring of SQL/Foundation.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

6.3 <cast specification>

Function

Specify a data conversion.

Format

No additional Format items.

Syntax Rules

- 1) Augments SR6 If the <cast operand> is a <value expression>, then the valid combinations of *TD* and *SD* in a <cast specification> are given by the following table.

<data type> <i>SD</i> of <value expression>	<data type> of <i>TD</i>																	
	EN	AN	VC	FC	VB	FB	D	T	TS	YM	DT	P	ET	BO	ADT	NT	CL	BL
EN	Y	Y	Y	Y	N	N	N	N	N	M	M	N	Y	N	M	M	Y	N
AN	Y	Y	Y	Y	N	N	N	N	N	N	N	N	N	N	M	M	Y	N
C	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	M	M	Y	N
B	N	N	Y	Y	Y	Y	N	N	N	N	N	N	N	N	M	M	Y	N
D	N	N	Y	Y	N	N	Y	N	Y	N	N	Y	N	N	M	M	Y	N
T	N	N	Y	Y	N	N	N	Y	Y	N	N	N	N	N	M	M	Y	N
TS	N	N	Y	Y	N	N	Y	Y	Y	N	N	Y	N	N	M	M	Y	N
YM	M	N	Y	Y	N	N	N	N	N	Y	N	N	N	N	M	M	Y	N
DT	M	N	Y	Y	N	N	N	N	N	N	Y	N	N	N	M	M	Y	N
P	N	N	Y	Y	N	N	Y	N	Y	N	N	Y	N	N	M	M	Y	N
ET	Y	N	Y	Y	N	N	N	N	N	N	N	N	Y	N	M	M	Y	N
BO	N	N	Y	Y	N	N	N	N	N	N	N	N	N	Y	M	M	Y	N
ADT	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M
NT	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	N	M	N
BL	N	N	N	N	N	N	N	N	N	N	N	N	N	N	M	N	N	Y

Where:

P = Period

Access Rules

No additional Access Rules.

General Rules

- 1) Insert this GR If *TD* is a period data type with element type *TED* and *SD* is not a period data type, then

Case:

- a) If *SD* is character string, then *SV* is replaced by:

TRIM (BOTH ' ' FROM *SV*)

Case:

6.3 <cast specification>

- i) If the rules for <literal> in Subclause 5.2, “<literal>”, can be applied to *SV* to determine a valid value of the data type *TD*, then *TV* is that value.
- ii) Otherwise, an exception condition is raised: *data exception — invalid character value for cast*.

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

```
PERIOD [ CAST ( SV AS TED ) , CAST ( SV AS TED ) )
```

- 2) Insert this GR If *TD* and *SD* are period data types with element types *TED* and *SED*, respectively, then *TV* is the result of:

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

```
PERIOD [ CAST ( BEGIN(SV) AS TED ) , CAST ( END(SV) AS TED ) )
```

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

```
PERIOD [ CAST ( FIRST (SV) AS TED ) , CAST ( NEXT ( LAST (SV) ) AS TED ) )
```

- 3) Insert this GR If *TD* is fixed-length character string, then let *LTD* be the length in characters of *TD*. If *TD* has a <character set specification>, then let *CSP* be that <character set specification>; otherwise, let *SCP* be a zero-length string. Let *TL* be the largest interger such that *TL**2 is less than *LTD*. *TV* is the result of:

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

```
CAST ( TRIM ( BOTH ' ' FROM CAST ( BEGIN (VE) AS CHARACTER(TL) CHARACTER SET CSP ) ) ||
' , ' || TRIM ( BOTH ' ' FROM CAST ( END (VE) AS CHARACTER(TL) CHARACTER SET CSP ) ) AS TD
)
```

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

```
CAST ( TRIM ( BOTH ' ' FROM CAST ( FIRST (VE) AS CHARACTER(TL) CHARACTER SET CSP ) ) ||
' , ' || TRIM ( BOTH ' ' FROM CAST ( NEXT ( LAST (VE) ) AS CHARACTER(TL) CHARACTER SET CSP
) ) AS TD )
```

- 4) Insert this GR If *TD* is varying-length character string, then let *MLTD* be the maximum length in characters of *TD*. If *TD* has a <character set specification>, then let *CSP* be that <character

set specification>; otherwise, let *SCP* be a zero-length string. Let *TL* be the largest integer such that $TL*2+1$ is less than *MLTD*. *TV* is the result of:

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

```
CAST ( TRIM ( BOTH ' ' FROM CAST ( BEGIN (VE) AS CHARACTER(TL) CHARACTER SET CSP ) ) ||  
' , ' || TRIM ( BOTH ' ' FROM CAST ( END (VE) AS CHARACTER(TL) CHARACTER SET CSP ) ) AS TD  
)
```

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

```
CAST ( TRIM ( BOTH ' ' FROM CAST ( FIRST (VE) AS CHARACTER(TL) CHARACTER SET CSP ) ) ||  
' , ' || TRIM ( BOTH ' ' FROM CAST ( NEXT ( LAST (VE) ) AS CHARACTER(TL) CHARACTER SET CSP  
) ) AS TD )
```

6.4 <value expression>

6.4 <value expression>**Function**

Specify a value.

Format

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

```
<value expression primary> ::=
    !! All alternatives from ISO/IEC 9075-2
    | <proximity function>
    | <period bound function>
    | <period duration function>
```

```
<proximity function> ::=
    { PRIOR | NEXT } <proximity argument>
```

```
<proximity argument> ::= <value expression>
```

```
<period bound function> ::=
```

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

```
{ BEGIN | END | LAST }
<left paren> <period value expression 1> <right paren>
```

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

```
{ FIRST | LAST } <left paren> <period value expression 1> <right paren>
```

```
<period value expression 1> ::= <period value expression>
```

```
<period duration function> ::=
    INTERVAL <left paren> <period value expression 2>
    [ <interval qualifier> ] <right paren>
```

```
<period value expression 2> ::= <period value expression>
```

Syntax Rules

- 1) Insert this SR The data type of <period value expression> shall be a period data type.
- 2) Insert this SR If <proximity function> is specified, then let the data type of <proximity argument> be *DP*. *DP* shall be a data type that is capable of being a period element type. The data type of the result is *DP*.
- 3) Insert this SR If <period bound function> is specified, then the data type of the result is the element type of <period value expression 1>.

- 4) Insert this SR If <period duration function> is specified, then let *EDT* be the element type of <period value expression 2>.
- a) If *EDT* is a datetime data type, then <interval qualifier> shall be specified; let *IQ* be the <interval qualifier>. Otherwise, let *IQ* be <space>.
 - b) Let *V1* and *V2* be two values of *EDT*. The data type of the result is the same as the data type of (*V1* – *V2*) *IQ*.

Access Rules

No additional Access Rules.

General Rules

- 1) Insert this GR If <proximity function> is specified, then let *DT* be the data type and let *VP* be the value of the <proximity argument> that it immediately contains.
- a) If *VP* is null, then the result is the null value. Otherwise,
Case:
 - i) If PRIOR is specified, then the result *VR* is that value of *DT* such that $VR < VP$ and there exists no value *NV* of *DT* such that $VR < NV < VP$.
 - ii) If NEXT is specified, then the result *VR* is that value of *DT* such that $VP < VR$ and there exists no value *NV* of *DT* such that $VP < NV < VR$.
 - b) If the type of the result is datetime and, after applying the previous subrule, any datetime field of the result is outside the permissible range of values for the field, or the result is invalid based on the natural rules for dates and times, then an exception condition is raised: *data exception — datetime field overflow*.
- 2) Insert this GR If a <period bound function> is specified, then let *P* be the result of <period value expression 1>. If *VP* is null, then the result is the null value. Otherwise, the result of the <period bound function> is

Case:

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

- a) If BEGIN is specified, then the beginning bound of *P*.
- b) If END is specified then the ending bound of *P*.
- c) If LAST is specified, then the result of

PRIOR (END (*P*))

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

- d) If FIRST is specified, then the beginning bound of *P*.

6.4 <value expression>

e) If LAST is specified, then the ending bound of P .

3) Insert this GR If a <period duration function> is specified, then let P be the result of <period value expression 2>. The result of the <period duration function> is the value of:

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

$(\text{END}(P) - \text{BEGIN}(P)) IQ$

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

$(\text{NEXT} (\text{LAST} (P)) - \text{FIRST} (P)) IQ$

6.5 <period value expression>

Function

Specify an expression whose data type is a period data type.

Format

```

<period value expression> ::=
    <period term>
    | <period value expression> P_UNION <period term>
    | <period value expression> P_EXCEPT <period term>
    | <period value constructor>

<period term> ::=
    <period primary>
    | <period term> P_INTERSECT <period primary>

<period primary> ::=
    <value expression primary>
    | <period arithmetic primary>

<period arithmetic primary> ::=
    <period value expression> <period arithmetic operator> <value expression>
    | <value expression> <period arithmetic operator> <period value expression>

<period arithmetic operator> ::=
    <plus sign>
    | <minus sign>

```

Syntax Rules

- 1) Let P_SETOP be P_UNION , P_EXCEPT , or $P_INTERSECT$.
- 2) The data type of the result of $P1 P_SETOP P2$ is determined by applying the rules of Subclause 10.3, “Set operation result data types”, to the data types of $P1$ and $P2$.
- 3) Let P be a <period value expression>, let V be a <value expression>, and let PAO be a <period arithmetic operator>. $P PAO V$ and $V PAO P$ are both equivalent to:

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

```
PERIOD [ BEGIN(P) PAO V , END(P) PAO V )
```

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

```
PERIOD [ FIRST ( P ) PAO V , NEXT ( LAST ( P ) ) PAO V )
```

- 4) The data type of <value expression primary> shall be a period data type.

6.5 <period value expression>

Access Rules

None.

General Rules

1) The result of *S1 P_SETOP S2* is determined as follows:

Let $MAX(X, Y)$ denote the greater of X and Y , and let $MIN(X, Y)$ denote the lesser of X and Y , where X and Y denote any two comparable values.

Case:

a) If *P_UNION* is specified, then

Case:

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

i) If *P1 OVERLAPS P2* is not true, then an exception condition is raised: *data exception — periods do not overlap*.

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

ii) If *P1 INTERSECTS P2* is not true, then an exception condition is raised: *data exception — periods do not intersect*.

iii) Otherwise the result of *P1 P_UNION P2* is:

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

`PERIOD [MIN (BEGIN (P1) , BEGIN (P2)) , MAX (END (P1) , END (P2)))`

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

`PERIOD [MIN (FIRST (P1) , FIRST (P2)) , MAX (LAST (P1) , LAST (P2))]`

b) If *P_EXCEPT* is specified, then,

Case:

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

-
- i) If $P1$ OVERLAPS $P2$ is not true, then the result is $P1$.

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

- ii) If $P1$ INTERSECTS $P2$ is not true, then the result is $P1$.

- iii) If $P1$ CONTAINS $P2$ OR $P2$ CONTAINS $P1$ is true, then an exception condition is raised: *data exception — invalid period*.

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

- iv) If $BEGIN(P2) < BEGIN(P1)$, then the result is:

PERIOD [MAX (BEGIN ($P1$) , END ($P2$)) , END ($P1$)]

- v) Otherwise the result is:

PERIOD [BEGIN ($P1$) , MIN (BEGIN ($P2$) , END ($P1$))]

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

- vi) If $FIRST(P2) < FIRST(P1)$, then the result is:

PERIOD [MAX (FIRST ($P1$) , LAST ($P2$)) , LAST ($P1$)]

- vii) Otherwise the result is:

PERIOD [FIRST ($P1$) , MIN (PRIOR (FIRST ($P2$)) , LAST ($P1$))]

- c) If P_INTERSECT is specified, then:

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

- i) If $P1$ OVERLAPS $P2$ is not true, then an exception condition is raised: *data exception — periods do not overlap*.

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

6.5 <period value expression>

- ii) If $P1$ INTERSECTS $P2$ is not true, then an exception condition is raised: *data exception — periods do not intersect.*
-

- iii) Otherwise the result of $P1$ P_INTERSECT $P2$ is:

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

PERIOD [MAX (BEGIN ($P1$) , BEGIN ($P2$)) , MIN (END ($P1$) , END ($P2$)))

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

PERIOD [MAX (FIRST ($P1$) , FIRST ($P2$)) , MIN (LAST ($P1$) , LAST ($P2$))]

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

6.6 <set or multiset value expression>

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

SQL/Foundation no longer defines a <set or multiset value expression>. See Possible Problem

TMPRL-019.

Function

Specify an expression whose result is of collection type SET or MULTISSET.

Format

```
<set or multiset value expression> ::=
    !! All alternatives from ISO/IEC 9075-2
    | <period set value function>
    | <expand function>

<expand function> ::=
    EXPAND <left paren> <period value expression> <right paren>
```

Syntax Rules

- 1) **Insert this SR** The data type of an <expand function> is SET and its element type is the element type of the immediately contained <period value expression> *PVE*.

Access Rules

No additional Access Rules.

General Rules

- 1) **Insert this GR** The result of <expand function> is a set whose elements are duplicates of all the elements of *PVE*, namely every value *V* of the element type of *PVE*, such that $BEGIN(PVE) \leq V \leq END(PVE)$. $FIRST(PVE) \leq V \leq LAST(PVE)$.

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

6.7 <period set value function>

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

SQL/Foundation no longer defines a <set or multiset value expression>. See Possible Problem

TMPRL-019.

Function

Specify a function returning a value of data type SET(PERIOD).

Format

```
<period set value function> ::=
    <normalize period set function>

<normalize period set function> ::=
    NORMALIZE <left paren> <period collection> <right paren>

<period collection> ::=
    <set or multiset value expression>
```

Syntax Rules

- 1) The element type *PCET* of a <period collection> shall be a period data type *PT*.
- 2) The data type of <normalize period set function> is SET(*PT*).

Access Rules

None.

General Rules

- 1) Let *PC* be the <period collection> specified in a <normalize period set function> *NPSF*. The value returned by *NPSF* is the result of the following <scalar subquery>:

```
( SELECT S_UNION (SET {P})
  FROM
    ( WITH PCT AS ( SELECT DISTINCT P FROM TABLE(PC) AS S(P) )
      SELECT DISTINCT PERIOD [ BEGIN(F.P) , END(L.P) ] AS P
    FROM PCT AS F, PCT AS L
    WHERE BEGIN(F.P) < END(L.P)
      AND NOT EXISTS ( SELECT *
                      FROM PCT AS M
                      WHERE BEGIN(F.P) < BEGIN(M.P)
                        AND BEGIN(M.P) < END(L.P)
                        AND NOT EXISTS ( SELECT *
                                      FROM PCT AS A1
```

SC32 N00651 = WG3:YYJ-007 = H2-2001-144
6.7 <period set value function>

```
WHERE BEGIN(A1.P) < BEGIN(M.P)
      AND BEGIN(M.P) <= END(A1.P) ) )
AND NOT EXISTS ( SELECT *
                  FROM PCT AS A2
                  WHERE ( BEGIN(A2.P) < BEGIN(F.P)
                          AND
                          BEGIN(F.P) <= END(A2.P) )
                        OR ( BEGIN(A2.P) <= END(L.P)
                          AND
                          END(L.P) < END(A2.P) ) )
) AS T )
```

6.8 <period value constructor>

Function

Specify a pair of values to be constructed into a period.

Format

```
<period value constructor> ::=
    PERIOD
        <beginning period delimiter> <beginning bound specification>
            <comma> <ending bound specification> <ending period delimiter>

<beginning bound specification> ::= <value expression>

<ending bound specification> ::= <value expression>
```

Syntax Rules

- 1) Let *BB* and *EB* be <beginning bound specification> and <ending bound specification>, respectively, and let their data types be *DTB* and *DTE* respectively. *DTB* and *DTE* shall be comparable.
- 2) The data type of <period value constructor> is period, with element type determined by applying Subclause 10.3, “Set operation result data types”, to *DTB* and *DTE*.
NOTE 8 – Valid element types are specified in Subclause 6.1, “<data type>”.

Access Rules

None.

General Rules

- 1) If the result of <beginning bound specification> or <ending bound specification> is null, then the result of the <period value constructor> is the null value.
- 2) The result of <period value constructor> is a period of which:
 - a) The beginning bound is

Case:

 - i) If <left bracket> is specified, then the value of <beginning bound specification>.
 - ii) Otherwise, the value of PRIOR (<beginning bound specification>).
 - b) The ending bound is:

Case:

 - i) If <right paren> is specified, then the value of <ending bound specification>.
 - ii) Otherwise the value of NEXT (<ending bound specification>).

- 3) If any specification or operation attempts to cause an item of type period to take a value where the beginning bound is greater than or equal to the ending bound, then an exception condition is raised: *data exception — invalid period*.

SC32 N00651 = WG3:YYJ-007 = H2-2001-144

7 Query expressions

7.1 <normalized query specification>

Function

Specify a table that is normalized with respect to one or more period columns.

Format

```
<normalized query specification> ::=
    <query specification> <normalize clause>

<normalize clause> ::=
    NORMALIZE ON <normalizing column reference list>

<normalizing column reference list> ::=
    <item reference> [ { <comma> <item reference> }... ]
```

Syntax Rules

- 1) Let *NQS* be the <normalized query specification>. Let *NCRL* be the <normalizing column reference list> immediately contained in the <normalize clause> immediately contained in *NQS*. Let *T* be the result of the <query specification> immediately contained in *NQS*.
- 2) Each <item reference> contained in *NCRL* shall be a column reference and shall unambiguously reference a column *C* of *T*. The data type of *C* shall be some period data type.

Access Rules

None.

General Rules

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

- 1) Let *N* be the number of <item reference>s contained in *NCRL*. Let CR_i , $1 \leq i \leq N$, be the *i*-th such <item reference>.
- 2) Let *TSL* be a <select list> that is a <value expression> sequence in which each <value expression> is a column reference that references a column of *T* and each column of *T* is referenced exactly once. The columns are referenced in the ascending sequence of their ordinal positions within *T*.

7.1 <normalized query specification>

- 3) Let *VES* be a <value expression> sequence (possibly empty) formed by deleting from *TSL* every column reference that references a column referenced by some *SR_i*. If *VES* is empty, then let *GBC* be the empty string; otherwise, let *GBC* be:

GROUP BY *VES*

- 4) The result of *NQS* is the result of

```

WITH T1 AS ( SELECT
NORMALIZE (SETUNION (SET{CR1})) AS CR1 ,
NORMALIZE (SETUNION (SET{CR1})) AS CR2 ,
.
.
.

NORMALIZE (SETUNION (SET{CR1})) AS CRN ,
VES
FROM T
GBC )
SELECT TSL
FROM T1 ,
TABLE(T1.CR1) AS TCR1(CR1)
TABLE(T1.CR2) AS TCR2(CR2)
.
.
.

TABLE(T1.CRN) AS TCRN(CRN)

```

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

- 5) Let *T2* be the table that results from applying the General Rules of Subclause 9.1, “Expansion of a table”, with *T* as *TABLE* and *NRCL* as *COLUMN REFERENCE LIST*, respectively.
- 6) Let *T3* be the table that results from applying the General Rules of Subclause 9.2, “Compression of a table”, with *T2* as *TABLE* and *NRCL* as *COLUMN REFERENCE LIST*, respectively.
- 7) The result of *NQS* is *T3*.
-

7.2 <query expression>

Function

Specify a table.

Format

```
<non-join query expression> ::=  
  <non-join query term>  
  | <query expression> UNION [ ALL | <expanding clause> ]  
    [ <corresponding spec> ] <query term>  
  | <query expression> EXCEPT [ ALL | <expanding clause> ]  
    [ <corresponding spec> ] <query term>  
  
<non-join query term> ::=  
  <non-join query primary>  
  | <query term> INTERSECT [ ALL | <expanding clause> ]  
    [ <corresponding spec> ] <query primary>  
  
<simple table> ::=  
  !! All alternatives from ISO/IEC 9075-2  
  | <normalized query specification>
```

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

```
<expanding clause> ::=  
  EXPANDING <left paren> <expanding column reference list> <right paren>  
  
<expanding column reference list> ::=  
  <item reference> [ { <comma> <item reference> }... ]
```

Syntax Rules

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

No additional Syntax Rules.

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

Insert this SR If <expanding clause> is specified, then:

- a) Let *ECRL* be the <expanding column reference list>.
- b) Each <column reference> contained in *ECRL* shall identify a column *C1* of *T1*. Each such column reference shall also implicitly identify some corresponding column *C2* of *T2*

7.2 <query expression>

- c) The data type of *C1* and the data type of *C2* shall be some period type.
-

Access Rules

No additional Access Rules.

General Rules

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

- 1) Insert this GR If <expanding clause> is specified, then let *OP* be whichever of UNION, EXCEPT, and INTERSECT is also specified. The result of the <query expression> is obtained by applying Subclause 7.3, “<expanding clause>”, in the context of *OP* with operands *T1* and *T2*.

NOTE 9 – “Context” is defined in Syntax Rule 1) of Subclause 7.3, “<expanding clause>”.

****Editor’s Note****

The preceding Syntax Rule uses the phrase "in the context of [UNION, EXCEPT, or INTERSECT]". Syntax Rule 1) of Subclause 7.3, “<expanding clause>”, defined "the context of [the <expanding clause>]", which is not the same thing at all. Surely that is not what the authors of DBL:MCI-067/X3H2-96-181 intended! I suspect that the problem is that the authors of DBL:MCI-067/X3H2-96-181 were merely careless about defining terms such as "context" and "specifying the context", etc. However, a correction is required here.

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

Insert before GR3 If <expanding clause> is specified, then:

- a) Let *ECRL* be the <expanding column reference list>.
- b) *T1* is replaced by the result of applying the General Rules of Subclause 9.1, “Expansion of a table”, with *T1* as *TABLE* and *ERCL* as *COLUMN REFERENCE LIST*, respectively.
- c) *T2* is replaced by the result of applying the General Rules of Subclause 9.1, “Expansion of a table”, with *T2* as *TABLE* and *ERCL* as *COLUMN REFERENCE LIST*, respectively.
- Insert after GR3 If <expanding clause> is specified, then *T* is replaced by the result of applying the General Rules of Subclause 9.2, “Compression of a table”, to *T* as *TABLE* and *ERCL* as *COLUMN REFERENCE LIST*, respectively.
-

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

7.3 <expanding clause>

Function

Qualify a table operation to operate on the expansion of its operand or operands and normalize the result.

Format

```
<expanding clause> ::=  
    EXPANDING <left paren> <expanding column reference list> <right paren>
```

```
<expanding column reference list> ::=  
    <item reference> [ { <comma> <item reference> }... ]
```

Syntax Rules

- 1) Let *EC* be the <expanding clause> and let *ECRL* be the <expanding column reference list>. Let the *context* of *EC* be an application of this Subclause, specifying a *table operator OP*, a *table operand T1*, and optionally a second *table operand T2*.
- 2) Each <item reference> contained in *ECRL* shall be a column reference and shall unambiguously reference a column *C1* of *T1*. If *T2* is specified, then each such column reference shall also implicitly identify some corresponding column *C2* of *T2* according to the Syntax Rules of the Subclause specifying the context of *EC*. The data type of *C1*, and the data type of *C2* if applicable, shall be some period data type.

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

The preceding Syntax Rule uses the phrase "of the Subclause specifying the context of [the <expanding clause>]". Syntax Rule 1) of this Subclause does that, so that phrase translates to "of Subclause 7.3, "<expanding clause>"". Surely that is not what the authors of DBL:MCI-067/X3H2-96-181 intended! I suspect that the problem is that the authors of DBL:MCI-067/X3H2-96-181 were merely careless about defining terms such as "context" and "specifying the context", etc. However, a correction is required here.

Access Rules

None.

General Rules

- 1) Let *n* be the number of <item reference>s contained in *ERCL*. Let *CR_i*, where *i* ranges from 1 (one) to *n*, be the *i*-th such <item reference>. Let *CM_i* be the <column name> of *CR_i*.

7.3 <expanding clause>

- 2) Let *TSL1* be a <select list> that is a <value expression> sequence in which each <value expression> is a column reference that references a column of *T1*, each column of *T1* is referenced exactly once, and the columns are referenced in the ascending sequence of their ordinal position within *T1*.
- 3) If *T2* is specified, then let *TSL2* be a <select list> that is a <value expression> sequence in which each <value expression> is a column reference that references a column of *T2*, each column of *T2* is referenced exactly once, and the columns are referenced in the ascending sequence of their ordinal position within *T2*.
- 4) Let *C1* be a <value expression> sequence, followed by a comma unless the sequence is empty, formed by deleting from *TSL1* every column reference that references a column referenced by some *CR_i*.
- 5) If *T2* is specified, then let *C2* be a <value expression> sequence, followed by a comma unless the sequence is empty, formed by deleting from *TSL2* every column reference that references a column referenced by some *CR_i*.

- 6) The *expansion* of *T1* is the result of

```

WITH E1 AS
( SELECT EXPAND(CR1) AS CN1 ,
  . . . ,
  EXPAND(CRn) AS CNn ,
  C1
FROM T1 )
SELECT C1 TCR1.CR1 , . . . , TCRn.CNn
FROM E1 , TABLE(E1.CR1) AS TCR1(CN1) ,
. . . , TABLE(E1.CRn) AS TCRn(CNn)

```

- 7) If *T2* is specified, then the *expansion* of *T1* is the result of

```

WITH E2 AS
( SELECT EXPAND(CR1) AS CN1 ,
  . . . ,
  EXPAND(CRn) AS CNn ,
  C2
FROM T2 )
SELECT C2 TCR1.CR1 , . . . , TCRn.CNn
FROM E2 , TABLE(E2.CR1) AS TCR1(CN1) ,
. . . , TABLE(E2.CRn) AS TCRn(CNn)

```

- 8) Let *ER* be the result of applying *OP* to the expansion of *T1* and, if applicable, the expansion of *T2*. Let *TEMP* be the result of

```

SELECT C1
PERIOD [ CN1 , CN1 ] AS CN1 ,
. . .
PERIOD [ CNn , CNn ] AS CNn ,
FROM ER
NORMALIZE ON CN1 , . . . , CNn

```

- 9) Let *OCL* be a list of *n* <column reference>s such that each column of *TEMP* is referenced by exactly one element of *OCL* and the *i*-th element corresponds to the *i*-th column of *T1*. The result of the application of *EC* in the context of *OP* and the given operand or operands is the result of

```

SELECT OCL
FROM TEMP

```

SC32 N00651 = WG3:YYJ-007 = H2-2001-144
7.3 <expanding clause>

SC32 N00651 = WG3:YYJ-007 = H2-2001-144

8 Predicates

8.1 <predicate>

Function

Specify a condition that can be evaluated to give a truth value of true, false, or unknown.

Format

```
<predicate> ::=
    !! All alternatives from ISO/IEC 9075-2
    | <period predicate>
```

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

- 1) Augments GR1 The result of a <predicate> is a truth value derived according to

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

the General Rules of Subclause 8.2, “<comparison predicate>”, Subclause 8.4, “<period predicate>”, or Subclause 8.3, “<overlaps predicate>”, as appropriate.

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

the General Rules of Subclause 8.2, “<comparison predicate>”, or Subclause 8.4, “<period predicate>”, as appropriate.

8.2 <comparison predicate>

Function

Specify a comparison of two row values.

Format

No additional Format items.

Syntax Rules

- 1) Insert this SR If any pair of respective values has a period data type, then their element types shall be comparable.

Access Rules

No additional Access Rules.

General Rules

- 1) Insert this GR The comparison of two periods is effectively computed as follows:
 - a) Let $P1$ be the first period and let $P2$ be the second period.
 - b) The result of $P1 = P2$ is the result of:

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

$$(\text{BEGIN}(P1) , \text{END}(P1)) = (\text{BEGIN}(P2) , \text{END}(P2))$$

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

$$(\text{FIRST} (P1) , \text{LAST} (P1)) = (\text{FIRST} (P2) , \text{LAST} (P2))$$

-
-
- c) The result of $P1 < P2$ is the result of:

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

$$(\text{BEGIN}(P1) , \text{END}(P1)) < (\text{BEGIN}(P2) , \text{END}(P2))$$

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

SC32 N00651 = WG3:YYJ-007 = H2-2001-144
8.2 <comparison predicate>

(FIRST (P1) , LAST (P1)) < (FIRST (P2) , LAST (P2))

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

8.3 <overlaps predicate>

Function

Specify a test for an overlap between two periods.

Format

```
<overlaps predicate> ::=
    <overlaps operand 1> OVERLAPS <overlaps operand 2>

<overlaps operand 1> ::= <overlaps operand>

<overlaps operand 2> ::= <overlaps operand>

<overlaps operand> ::=
    <row value expression>
    | <period value expression>
```

Syntax Rules

- 1) Replaces SR1) and SR2) The degree of <row value expression> shall be 2. Let the two elements be *E1* and *E2* respectively and let their data types be *DT1* and *DT2* respectively. *DT1* shall be a datetime data type and *DT2* shall be either a datetime data type or an interval data type such that *E1* + *E2* is a valid <datetime value expression>.
- 2) Replaces SR3) Case:
 - a) If *DT2* is INTERVAL, then *E1* shall be a valid <datetime value expression>. <row value expression> is equivalent to:

$$\text{PERIOD} (E1 , E1 + E2)$$
 - b) Otherwise, *DT1* and *DT2* shall be comparable and the data type of <overlaps operand> is period with element type determined by applying Subclause 10.3, “Set operation result data types”, to *DT1* and *DT2*.

Access Rules

No additional Access Rules.

General Rules

- 1) Replaces GR1) Let *D* and *E* be the values of the elements of <row value expression>.
- 2) Replaces GR3) and GR5) If *D* is the null value or if *E* < *D*, then let *L* = *E* and let *U* = *D*; otherwise, let *L* = *D* and let *U* = *E*.

- 3) Replaces GR2 The value of <overlaps operand> is a period whose beginning bound is L and whose ending bound is U .
 - 4) Replaces GR4 Let $P1$ and $P2$ be the values of <overlaps operand 1> and <overlaps operand 2>, respectively.
 - 5) Replaces GR6 The result of <overlaps predicate> is the result of:
NOT ($P1$ PRECEDES $P2$
OR $P1$ SUCCEEDS $P2$
OR $P1$ MEETS $P2$)
-

8.4 <period predicate>

8.4 <period predicate>

Function

Specify a comparison of two periods or element values.

Format

```
<period predicate> ::=
    <period or element 1> <period comparison op> <period or element 2>
```

```
<period comparison op> ::=
    PRECEDES
    | SUCCEEDS
    | MEETS
    | CONTAINS
```

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

```
| INTERSECTS
```

```
<period or element 1> ::= <period or element argument>
```

```
<period or element 2> ::= <period or element argument>
```

```
<period or element argument> ::=
    <period value expression>
    | <value expression>
```

Syntax Rules

- 1) Let *PD1* be <period or element 1> and let *PD2* be <period or element 2>.
- 2) If *PD1* is a <period value expression>, then let *DT1* be the data type of *BEGIN(PD1)*; otherwise, let *DT1* be the data type of *PD1*. *DT1* shall be a data type that is capable of being a period element type.
- 3) If *PD2* is a <period value expression>, then let *DT2* be the data type of *BEGIN(PD2)*; otherwise, let *DT2* be the data type of *PD2*. *DT2* shall be a data type that is capable of being a period element type.
- 4) *DT1* and *DT2* shall be comparable.

Access Rules

None.

General Rules

- 1) If *PD1* is a <period value expression>, then let *P1* be *PD1*; otherwise,
Case:
 - a) If *PD1* is the null value, then let *P1* be the null value.
 - b) Otherwise, let *P1* be PERIOD [*PD1* , *PD1*].
- 2) If *PD2* is a <period value expression>, then let *P2* be *PD2*; otherwise,
Case:
 - a) If *PD2* is the null value, then let *P2* be the null value.
 - b) Otherwise, let *P2* be PERIOD [*PD2* , *PD2*].
- 3) Let *P_COMPOP* be the specified <period comparison op>. The result of *P1 P_COMPOP P2* is determined as follows:
 - a) If *P_COMPOP* is PRECEDES, then the result is:

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

END (*P1*) < BEGIN (*P2*)

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

LAST (*P1*) < FIRST (*P2*)

-
-
- b) If *P_COMPOP* is SUCCEEDS, then the result is:

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

BEGIN (*P1*) > END (*P2*)

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

FIRST (*P1*) > LAST (*P2*)

-
-
- -
 - c) If *P_COMPOP* is MEETS, then the result is:

8.4 <period predicate>

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

END (P1) = BEGIN (P2) OR

BEGIN (P1) = END (P2)

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

NEXT (LAST (P1)) = FIRST (P2) OR

FIRST (P1) = NEXT (LAST (P2))

d) If *P_COMPOP* is CONTAINS, then the result is:

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

BEGIN (P1) <= BEGIN (P2) AND

END (P1) >= END (P2)

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

FIRST (P1) <= FIRST (P2) AND

LAST (P1) >= LAST (P2)

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

4) If *P_COMPOP* is INTERSECTS, then the result is:

LAST (P1) >= FIRST (P2) AND

LAST (P2) >= FIRST (P1)

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

9 Expansion and compression

9.1 Expansion of a table

Function

Specify rules for expanding a table having one or more columns of period data type with respect to one or more of those columns.

Syntax Rules

- 1) Let T be a *TABLE* and CRL be a *COLUMN REFERENCE LIST* specified in an application of this Subclause.
- 2) Every <column reference> in CRL shall reference a column of T and the data type of each column so referenced shall be a period type.

General Rules

- 1) Let R be a row in T . The *expanded period row set* EPS of R is a set of rows of the same row type as R . A row $EPSR$ is a member of EPS if and only if all of the following are true:
 - a) R and $EPSR$ are value equivalent with respect to every column of T that is not identified by a member of CRL .
 - b) For every column C of T that is identified by a member of CRL , the value for C in $EPSR$ is a period P such that:

$$\text{FIRST}(R.C) \leq \text{FIRST}(P) = \text{LAST}(P) \leq \text{LAST}(C)$$
- 2) The result of expanding T with respect to CRL is the table that consists of every row that is a member of the expanded period row set of some row in T .

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

9.2 Compression of a table

Function

Specify rules for compressing a table having one or more columns of period data type with respect to one or more of those columns.

Syntax Rules

- 1) Let T be a *TABLE* and CRL be a *COLUMN REFERENCE LIST* specified in an application of this Subclause.
- 2) Every <column reference> in CRL shall reference a column of T and the data type of each column so referenced shall be a period type.

General Rules

- 1) Let N be the number of <column reference>s contained in CRL . Let CR_i , $1 \text{ (one)} \leq i \leq N$, be the i -th such <column reference>.
 - 2) Let $T2$ be a copy of $T1$. For i ranging from 1 (one) to N , while there exists a pair of rows ($R1$, $R2$) in $T2$ that are value-equivalent with respect to CR_i and $\text{MEETS}(R1.CR_i, R2.CR_i)$:
 - a) A row is inserted into $T2$ that is value-equivalent to $R1$ with respect to CR_i and, for CR_i has the value $\text{P_UNION} (R1.CR_i, R2.CR_i)$.
 - b) $R1$ and $R2$ are deleted from $T2$.
 - 3) The result of compressing $T1$ with respect to CRL is $T2$.
-

10 Additional common rules

10.1 Retrieval assignment

Function

Specify rules for value assignments that retrieve SQL-data.

Syntax Rules

- 1) Augments SR2 If the data type of T is character string, bit string, numeric, datetime, interval, or period, then the data type of V shall be a mutually assignable character string type, a bit string type, a numeric type, the same datetime type, a comparable interval type, or the same period type, respectively.

Access Rules

No additional Access Rules.

General Rules

- 1) Insert this GR If the data type of T is period and there is a representation of the value of V in the data type of T , then the value of T is set to that representation.

10.2 Store assignment

Function

Specify rules for value assignments that store SQL-data.

Syntax Rules

- 1) Augments SR2 If the data type of T is character string, bit string, numeric, datetime, interval, or period, then the data type of V shall be a mutually assignable character string type, a bit string type, a numeric type, the same datetime type, a comparable interval type, or the same period type, respectively.

Access Rules

No additional Access Rules.

General Rules

- 1) Insert this GR If the data type of T is period and there is a representation of the value of V in the data type of T , then the value of T is set to that representation.

10.3 Set operation result data types

Function

Specify the Syntax Rules and result data types for <case expression>s and <query expression>s having set operators.

Syntax Rules

1) Augments SR3 Case:

- a) If any data type in *DTS* is PERIOD(*ET*) for some data type *ET*, then every data type in *DTS* shall be a period data type, and the result is a period data type whose element type is determined by applying Subclause 10.3, “Set operation result data types”, to the element types of all the data types in *DTS*.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

SC32 N00651 = WG3:YYJ-007 = H2-2001-144

11 Schema definition and manipulation

11.1 <default clause>

Function

Specify the default for a column or domain.

Format

No additional Format items.

Syntax Rules

- 1) Insert this SR If a <literal> is specified and the subject data type is period, then the <literal> shall be a <period literal> and shall be of the same <period precision> as the subject data type.

Access Rules

No additional Access Rules.

General Rules

- 1) Insert this GR If the subject data type is period, then the default value inserted in the column descriptor, if the <default clause> is to apply to a column, is the value of the <literal>.

SC32 N00651 = WG3:YYJ-007 = H2-2001-144

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

SC32 N00651 = WG3:YYJ-007 = H2-2001-144

12 SQL-client modules

12.1 Data type correspondences

Function

Specify the data type correspondences for SQL data types and host language types.

Tables

Table 2—Data type correspondences for Ada

SQL Data Type	Ada Data Type
<i>All alternatives from ISO/IEC 9075-2</i>	
PERIOD	<i>None</i>

Table 3—Data type correspondences for C

SQL Data Type	C Data Type
<i>All alternatives from ISO/IEC 9075-2</i>	
PERIOD	<i>None</i>

Table 4—Data type correspondences for COBOL

SQL Data Type	COBOL Data Type
<i>All alternatives from ISO/IEC 9075-2</i>	
PERIOD	<i>None</i>

Table 5—Data type correspondences for Fortran

SQL Data Type	Fortran Data Type
<i>All alternatives from ISO/IEC 9075-2</i>	
PERIOD	<i>None</i>

Table 6—Data type correspondences for MUMPS

SQL Data Type	MUMPS Data Type
<i>All alternatives from ISO/IEC 9075-2</i> PERIOD	<i>None</i>

Table 7—Data type correspondences for Pascal

SQL Data Type	Pascal Data Type
<i>All alternatives from ISO/IEC 9075-2</i> PERIOD	<i>None</i>

Table 8—Data type correspondences for PL/I

SQL Data Type	PL/I Data Type
<i>All alternatives from ISO/IEC 9075-2</i> PERIOD	<i>None</i>

SC32 N00651 = WG3:YYJ-007 = H2-2001-144

13 Dynamic SQL

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

The treatment of period types in Dynamic SQL has been noted as a Possible Problem (see Possible Problem **TMPRL-015**), and Mike Sykes suggests that this Clause should be deleted, if only temporarily, perhaps leaving the Clause and Subclause headings as placeholders.

13.1 Description of SQL item descriptor areas

Function

Specify the identifiers, data types, and codes used in SQL item descriptor areas.

Syntax Rules

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

- 1) **Insert after SR6)m** TYPE indicates a <period type>, DATETIME_INTERVAL_CODE is a code specified in Table 10, “Codes associated with datetime and period data types in Dynamic SQL”, and PRECISION is a valid value for the <period precision> of the indicated period data type.

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

- 2) **Insert after SR4)b** If *DT* is a period type, then the descriptor of the associated element type of *DT*.
-

Access Rules

No additional Access Rules.

General Rules

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

- 1) **Augments GR1** Table 9, “Codes used for SQL data types in Dynamic SQL”, specifies the codes associated with the SQL data types.

13.1 Description of SQL item descriptor areas

Table 9—Codes used for SQL data types in Dynamic SQL

Data Type	Code
<i>All alternatives from ISO/IEC 9075-2</i> PERIOD	18

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

- 2) Augments GR2 Table 10, “Codes associated with datetime and period data types in Dynamic SQL”, specifies the codes associated with the datetime data types.

Table 10—Codes associated with datetime and period data types in Dynamic SQL

Datetime Data Type	Code
DATE	1
TIME	2
TIMESTAMP	3
TIME WITH TIME ZONE	4
TIMESTAMP WITH TIME ZONE	5
PERIOD	6
PERIOD WITH TIME ZONE	7

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

13.2 <get descriptor statement>

Function

Get information from an SQL descriptor area.

Format

No additional Format items.

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

- 1) Insert after GR5)a) If *IDA* is subordinate to an item descriptor area whose *TYPE* field indicates *PERIOD*, then an exception condition is raised: *dynamic SQL error — undefined DATA value*.

13.3 <set descriptor statement>

Function

Set information in an SQL descriptor area.

Format

No additional Format items.

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

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

- 1) Insert after GR4)d If *DIN* is TYPE and *V* indicates PERIOD or PERIOD WITH TIME ZONE, then PRECISION is set to 6 and all other item descriptor area fields are set to implementation-dependent values.

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

- 2) Insert after GR4)a)i If *IDA* is subordinate to an item descriptor area whose TYPE field indicates PERIOD, then an exception condition is raised: *dynamic SQL error — invalid DATA target*.
 - 3) Replace GR4)b)ii)1 If $V = K+1$ and TYPE in *PIDA* does not indicate ROW, ARRAY, ARRAY LOCATOR, or PERIOD, then an exception condition is raised: *dynamic SQL error — invalid LEVEL value*.
-

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

13.4 <prepare statement>

Function

Prepare a statement for execution.

Format

No additional Format items.

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

- 1) Insert after GR6)a) xv) 1) A) If *CT* immediately contains PERIOD, then *RT* is undefined.

13.5 <describe statement>

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 elements.

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

- 1) Augments GR8)d)vii) If TYPE indicates a <period type>, then LENGTH is set to the length in positions of the period type, DATETIME_INTERVAL_CODE is set to a code as specified in Table 10, “Codes associated with datetime and period data types in Dynamic SQL”, to indicate the specified period data type, and PRECISION is set to the <period precision>, if applicable.
- 2) Insert before GR8)d)viii) If TYPE indicates a <period type>, then LENGTH is set to the length in positions of the period type, DATETIME_INTERVAL_CODE is set to a code as specified in Table 10, “Codes associated with datetime and period data types in Dynamic SQL”, to indicate the specified period data type, and PRECISION is set to the <period precision>, if applicable.

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

14 Call-Level Interface specifications

14.1 Implicit EXECUTE USING and OPEN USING clauses

Function

Specify the rules for an implicit EXECUTE USING clause and an implicit OPEN USING clause.

General Rules

- 1) Replace GR4))ii)1) The value of LEVEL is zero and TYPE indicates ROW, ARRAY, or PERIOD.

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

14.2 Description of CLI item descriptor areas

Function

Specify the identifiers, data types and codes for fields used in CLI item descriptor areas.

Syntax Rules

- 1) Insert after SR3)b) If *DT* is PERIOD, then the descriptor of the associated element type of *DT*.
- 2) Replace SR7)b)ii) TYPE is none of ROW, ARRAY, or PERIOD.
- 3) Replace SR12)a)i) TYPE is ROW, ARRAY, or PERIOD.
- 4) Insert before SR12)c)x) TYPE indicates PERIOD, there is exactly 1 (one) immediately subordinate descriptor area of *IDA*, and that item descriptor area is valid.
- 5) Replace SR13)a)i) TYPE is ROW, ARRAY, or PERIOD.
- 6) Insert before SR13)c)vii) TYPE indicates PERIOD, there is exactly 1 (one) immediately subordinate descriptor area of *IDA*, and that item descriptor area is valid.

General Rules

Table 11—Codes used for implementation data types in SQL/CLI

Data Type	Code
PERIOD	18

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

14.3 SQL/CLI data type correspondences

Function

Specify the SQL/CLI data type correspondences for SQL data types and host language types associated with the required parameter mechanisms, as shown in Table 3, "Supported calling conventions of SQL/CLI routines by language", in ISO/IEC 9075-3.

Tables

Table 12—SQL/CLI data type correspondences for Ada

SQL Data Type	Ada Data Type
PERIOD	<i>None</i>

Table 13—SQL/CLI data type correspondences for C

SQL Data Type	C Data Type
PERIOD	<i>None</i>

Table 14—SQL/CLI data type correspondences for COBOL

SQL Data Type	COBOL Data Type
PERIOD	<i>None</i>

Table 15—SQL/CLI data type correspondences for Fortran

SQL Data Type	Fortran Data Type
PERIOD	<i>None</i>

Table 16—SQL/CLI data type correspondences for MUMPS

SQL Data Type	MUMPS Data Type
PERIOD	<i>None</i>

Table 17—SQL/CLI data type correspondences for Pascal

SQL Data Type	Pascal Data Type
PERIOD	<i>None</i>

14.3 SQL/CLI data type correspondences

Table 18—SQL/CLI data type correspondences for PL/I

SQL Data Type	PL/I Data Type
PERIOD	<i>None</i>

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

15 SQL/CLI routines

15.1 GetData

Function

Retrieve a column value.

Definition

No new Definition items.

General Rules

- 1) Replace GR11
 - 2) If DATA_POINTER is non-zero for at least one of the first N item descriptor areas of *ARD* for which LEVEL is 0 (zero) and the value of TYPE is none of ROW, ARRAY, and PERIOD, then let *BCN* be the column number associated with such an item descriptor area and let *HBCN* be the value of MAX(*BCN*). Otherwise, let *HBCN* be zero.
 - 3) Replace GR12 Let *IDA* be the item descriptor area of *ARD* specified by *CN*. If the value of TYPE in *IDA* is any of ROW, ARRAY, or PERIOD, or if the LEVEL of *IDA* is greater than 0 (zero), then an exception condition is raised: *dynamic SQL error — invalid descriptor index*.
-

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

15.2 GetParamData

Function

Retrieve the value of a dynamic output parameter.

Definition

No new Definition items.

General Rules

- 1) Replace GR8 If `DATA_POINTER` is non-zero for at least one of the first N item descriptor areas of *APD* for which the `TYPE` value is none of `ROW`, `ARRAY`, or `PERIOD`, then let BPN be the parameter number associated with such an item descriptor area and let $HBPN$ be the value of $\text{MAX}(BPN)$. Otherwise, let $HBPN$ be 0 (zero).
- 2) Replace GR9 Let *IDA* be the item descriptor area of *APD* specified by PN . if the value of `TYPE` of *IDA* is none of `ROW`, `ARRAY`, or `PERIOD`, or if `LEVEL` of *IDA* is greater than 0 (zero), then an exception condition is raised: *dynamic SQL error — invalid descriptor index*.

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

15.3 SetDescField

Function

Set a field in a CLI descriptor area.

Definition

No additional Definition items.

General Rules

- 1) Replace GR14)b)j) If Value is $K+1$ and TYPE in *PIDA* does not indicate ROW, ARRAY, ARRAY LOCATOR, or PERIOD, then an exception condition is raised: *dynamic SQL error — invalid LEVEL value.*

SC32 N00651 = WG3:YYJ-007 = H2-2001-144

16 Information Schema

16.1 ELEMENT_TYPES view

Function

Identify the array and period element types defined in this catalog that are accessible to a given user.

Definition

No additional Definition items

Conformance Rules

No additional Conformance Rules.

SC32 N00651 = WG3:YYJ-007 = H2-2001-144

17 Definition Schema

17.1 DATA_TYPE_DESCRIPTOR base table

Function

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

The DATA_TYPE_DESCRIPTOR table has one row for each domain and one row for each column (in each table) that is defined as having a data type rather than a domain. It effectively contains a representation of the data type descriptors.

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

The DATA_TYPE_DESCRIPTOR table has one row for each domain, one row for each column (in each table) and for each attribute (in each structured type) that is defined as having a data type rather than a domain, one row for each distinct type, one row for the result type of each SQL-invoked function, one row for each SQL parameter of each SQL-invoked routine, one row for the result type of each method specification, one row for each parameter of each method specification, and one row for each structured type whose associated reference type has a user-defined representation. It effectively contains a representation of the data type descriptors.

Definition

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

```
CREATE TABLE DATA_TYPE_DESCRIPTOR
(
  Note: A modification is required to the DATA_TYPE_DESCRIPTOR
  base table; this modification will be easier to specify when the
  definition of that table in SQL/Foundation has been modified to accommodate
  collection types.
)
```

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

SC32 N00651 = WG3:YYJ-007 = H2-2001-144
17.1 DATA_TYPE_DESCRIPTOR base table

Augment the column constraint DATA_TYPE_DESCRIPTOR_DATA_TYPE_CHECK_COMBINATIONS in Part 11

Add the following immediately preceding DATA_TYPE NOT IN:

```
( DATA_TYPE = 'PERIOD'  
AND  
  ( CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,  
    COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME ) IS NULL  
AND  
  ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX ) IS NULL  
AND  
  NUMERIC_SCALE IS NULL  
AND  
  DATETIME_PRECISION IS NULL  
AND  
  ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL  
AND  
  ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,  
    USER_DEFINED_TYPE_NAME ) IS NULL  
AND  
  ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL  
AND  
  MAXIMUM_CARDINALITY IS NULL )
```

Augment the column constraint DATA_TYPE_DESCRIPTOR_DATA_TYPE_CHECK_COMBINATIONS in Part 11

Add 'PERIOD', immediately preceding 'REF', 'ARRAY', 'ROW'.

Description

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

No additional Description items.

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

- 1) Insert after Description 10)) If DATA_TYPE is 'PERIOD', then the data type being described is a period type.
-

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

17.2 ELEMENT_TYPES base table

Function

The ELEMENT_TYPES table has one row for each array or period type. It effectively contains a representation of the element descriptor of the array type.

Definition

No additional Definition elements

Description

- 1) Replace Description 1) The values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE, and ARRAY_IDENTIFIER are the values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE, and DTD_IDENTIFIER respectively of the row in DATA_TYPE_DESCRIPTOR that describes the array or period type whose element type is being described.
- 2) Replace Description 2) The values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE, and DTD_IDENTIFIER are the values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE, and DTD_IDENTIFIER, respectively, of the row in DATA_TYPE_DESCRIPTOR that describes the element type of the array or period type.

SC32 N00651 = WG3:YYJ-007 = H2-2001-144

18 Status codes

18.1 SQLSTATE

The exception codes associated with the SQLSTATE parameter are extended to support the period data type.

Table 19—SQLSTATE class and subclass values

Category	Condition	Class	Subcondition	Subclass
X	<i>All alternatives from ISO/IEC 9075-2</i> data exception	22	(no subclass)	000
			invalid period format	032
			invalid period	033

SC32 N00651 = WG3:YYJ-007 = H2-2001-144

19 Conformance

New paragraph Claims of conformance to this Part of this International Standard shall state:

- 1) A conformance claim to Part 2 of this International Standard, in Clause 24, "Conformance".
- 2) The definitions for all elements and actions that this Part of ISO/IEC 9075 specifies as implementation-defined.
- 3) Other items as required.

SC32 N00651 = WG3:YYJ-007 = H2-2001-144

Annex A (informative)

Implementation-defined elements

New paragraph This Annex references those features that are identified in the body of this part of this International Standard as implementation-defined.

Insert after second paragraph The term *implementation-defined* is used to identify characteristics that may differ between implementations, but that shall be defined for each particular implementation.

There are no additional implementation-defined elements.

SC32 N00651 = WG3:YYJ-007 = H2-2001-144

Annex B (informative)

Implementation-dependent elements

New paragraph This Annex references those places where this part of this International Standard states explicitly that the actions of a conforming implementation are implementation-dependent.

Insert after second paragraph The term *implementation-dependent* is used to identify characteristics that may differ between implementations, but that are not necessarily specified for any particular implementation.

Subclause 13.3, “<set descriptor statement>”:

- 1) If *V* indicates PERIOD or PERIOD WITH TIME ZONE, then PRECISION is set to 6 and all other item descriptor area fields are set to implementation-dependent values.

SC32 N00651 = WG3:YYJ-007 = H2-2001-144

Annex C (informative)

Deprecated features

New paragraph It is intended that the following features will be removed at a later date from a revised version of this part of this International Standard:

There are no additional deprecated items.

SC32 N00651 = WG3:YYJ-007 = H2-2001-144

Annex D (informative)

Incompatibilities with ISO/IEC 9075:1999

New paragraph This part of this International Standard introduces some incompatibilities with the earlier version of Database Language SQL as specified in ISO/IEC 9075:1992. Unless specified in this Annex, features and capabilities of Database Language SQL are compatible with the earlier version of this International Standard.

1) A number of additional <reserved word>s have been added to the language. These <reserved word>s are:

- CONTAINS
- EXPAND
- EXPANDING

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

- INTERSECTS
-
- MEETS
 - NORMALIZE
 - PERIOD
 - PRECEDES
 - SUCCEEDS

SC32 N00651 = WG3:YYJ-007 = H2-2001-144

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

SC32 N00651 = WG3:YYJ-007 = H2-2001-144

Annex E (informative)

Typical header files

E.1 C header file SQLCLI.H

```
/* SQL data type codes */
#define SQL_PERIOD 18
```

E.2 COBOL library item SQLCLI

```
* SQL DATA TYPE CODES
01 SQL-PERIOD PIC S9(4) BINARY VALUE IS 18.
```

Annex F
(informative)

SQL Feature Taxonomy

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

To be supplied.

Index

Index entries appearing in **boldface** indicate the page where the word, phrase, or BNF nonterminal was defined; index entries appearing in *italics* indicate a page where the BNF nonterminal was used in a Format; and index entries appearing in roman type indicate a page where the word, phrase, or BNF nonterminal was used in a heading, Function, Syntax Rule, Access Rule, General Rule, Leveling Rule, Table, or other descriptive text.

— B —

beginning bound • 5, 16, 17, 31, 40, 41, 55
<beginning bound specification> • **40**
<beginning period delimiter> • **23, 40**
<beginning value> • **23**

— C —

<case expression> • 63
<cast specification> • 27
<column reference> • 45, 48, 59, 60
<comma> • 23, 40, 43, 45, 47
CONTAINS • 21, 35, 56, 58, 101
<corresponding spec> • 45

— D —

<data type> • 25, 27
<datetime value expression> • 54
<date value> • 23, 24
<delimiter token> • **21**

— E —

Editor's Note • 26, 37, 38, 46, 47, 71
element type • 5, 15, 16, 17, 19, 27, 28, 30, 31, 37, 38, 40, 52, 54, 56, 63, 71, 85, 89
ending bound • 5, 16, 17, 31, 32, 40, 41, 55
<ending bound specification> • **40**
<ending period delimiter> • **23, 40**
<ending value> • **23**
EXPAND • 21, 37, 48, 101
<expand function> • **37**
EXPANDING • 21, 45, 47, 101
<expanding clause> • 18, **45, 46, 47**
<expanding column reference list> • **45, 46, 47**

— G —

<general literal> • **23**
granule • 5, 15, 16

— I —

intersects • 17, 18
INTERSECTS • 21, 34, 35, 36, 56, 58, 101
<interval qualifier> • 30, 31
invalid period format • 91

<item reference> • 43, 45, 47

— L —

last element • 5, 16, 17
<left bracket> • 23, 40
<left paren> • 23, 25, 30, 37, 38, 45, 47

— M —

meets • 16, 18
MEETS • 21, 55, 56, 57, 60, 101
<minus sign> • 23, 33
<modified item> • **7**

— N —

<non-join query expression> • **45**
<non-join query primary> • 45
<non-join query term> • **45**
<non-reserved word> • **21**
NORMALIZE • 16, 21, 38, 43, 44, 48, 101
<normalize clause> • 16, 18, **43**
<normalized query specification> • 18, **43, 45**
<normalize period set function> • **38**
<normalizing column reference list> • 18, **43**

— O —

<overlaps operand 1> • **54, 55**
<overlaps operand 2> • **54, 55**
<overlaps operand> • **54, 55**
<overlaps predicate> • **54, 55**

— P —

Part 1 • 3, 6, 88
Part 10 • 3
Part 11 • 3, 88
Part 13 • 3
Part 14 • 3
Part 2 • 3, 6, 93
Part 3 • 3
Part 4 • 3
Part 9 • 3

SC32 N00651 = WG3:YYJ-007 = H2-2001-144

period • 1, 5, 9, 10, 11, 15, 16, 17, 18, 19, 21, 23, 24, 25, 26, 27, 28, 30, 31, 32, 33, 34, 35, 36, 37, 38, 40, 41, 43, 46, 47, 51, 52, 54, 55, 56, 57, 59, 60, 61, 62, 63, 65, 71, 72, 76, 85, 88, 89, 91

PERIOD • 1, 5, 15, 16, 19, 22, 23, 24, 25, 28, 33, 34, 35, 36, 38, 40, 48, 54, 57, 63, 68, 69, 72, 73, 74, 75, 77, 78, 79, 80, 81, 82, 83, 88, 97, 101, 104

<period arithmetic operator> • **33**

<period arithmetic primary> • **33**

<period bound function> • 17, **30**, 31

<period bound separator> • **23**

<period collection> • **38**

<period comparison op> • **56**, 57

<period duration function> • 17, **30**, 31, 32

<period literal> • **23**, 24, 65

<period or element 1> • **56**

<period or element 2> • **56**

<period or element argument> • **56**

<period predicate> • 51, **56**

<period primary> • **33**

<period set value function> • 37, **38**

<period term> • **33**

<period type> • **25**, 71, 76

<period value constructor> • 33, **40**

<period value expression 1> • **30**, 31

<period value expression 2> • **30**, 31, 32

<period value expression> • 30, **33**, 37, 54, 56, 57

<plus sign> • 33

precedes • 18

PRECEDES • 22, 55, 56, 57, 101

<predicate> • **51**

<proximity argument> • **30**, 31

<proximity function> • 17, **30**, 31

— Q —

<query expression> • 18, 45, 46, 63

<query primary> • 45

<query term> • 45

<quote> • 23

— R —

<reserved word> • **21**, 101

<right bracket> • 23

<right paren> • 23, 25, 30, 37, 38, 40, 45, 47

<row value expression> • 54

— S —

<scalar subquery> • 38

<set or multiset value expression> • **37**, 38

<simple table> • **45**

<space> • 23, 31

SUCCEEDS • 22, 55, 56, 57, 101

— T —

<time value> • 23, 24

<time zone interval> • 23

— V —

<value expression> • 27, **30**, 33, 40, 43, 44, 48, 56

<value expression primary> • **30**, 33

1 Possible problems with SQL/Temporal

I observe some possible problems with SQL/Temporal as defined in this document. These are noted below. Further contributions to this list are welcome. Deletions from the list (resulting from change proposals that correct the problems or from research indicating that the problems do not, in fact, exist) are even more welcome. Other comments may appear in the same list.

Because of the highly dynamic nature of this list (problems being removed because they are solved, new problems being added), it has become rather confusing to have the problem numbers automatically assigned by the document production facility. In order to reduce this confusion, I have instead assigned "fixed" numbers to each possible problem. These numbers will not change from printing to printing, but will instead develop "gaps" between numbers as problems are solved.

Possible problems related to SQL/Temporal

Significant Possible Problems:

999 In the body of the Working Draft, I have occasionally highlighted a point that requires urgent attention thus:

Editor's Note
Text of the problem.

These items are indexed under "***Editor's Note***".

TMPRL-022 The following Possible Problem has been noted:

Severity: Major Technical
Reference: P07, SQL/Temporal, No specific location
Note at: None
Source: WG3:RTM-073/X3H2-99-???
Possible Problem:

SQL/Temporal currently has no conformance rules and, *a fortiori*, no Annex containing a summary thereof. Nor has anything yet been done about the possible division of SQL/Temporal into features.

Proposed Solution:

None provided with comment.

Editor's Notes for WG3:YYJ-007 = H2-2001-144

Minor Problems and Wordsmithing Candidates:

Language Opportunities

TMPRL-023 The following Language Opportunity has been noted:

Severity: Language Opportunity

Reference: P02-19.06 (SQL/Foundation, Subclause 19.6, "<prepare statement>")

Note at: None

Source: WG3:SAF-029R1

Language Opportunity:

Paul Cotton wrote in a private comment on WG3:SAF-029R1:

The Rationale in WG3:SAF-029R1 states:

Conversely, a period column can be inserted or updated by using a <period value constructor>, vis.:

```
SET period-column = PERIOD (:first_value, :last_value]
```

But there are no Dynamic SQL rules to determine the data type for the <period value constructor PERIOD[?,5], where the <data type> of ? can be inferred from the <data type> of the other value in the constructor. Note that this problem does not exist for <array value constructor>s, since SQL/Foundation Subclause 19.6, "<prepare statement>", GR 6)a)xiv) covers this case. It is possible that there are other PERIOD functions that need to be covered.

Proposed Solution:

None provided with comment.

Index

Index entries appearing in **boldface** indicate the page where the word, phrase, or BNF nonterminal was defined; index entries appearing in *italics* indicate a page where the BNF nonterminal was used in a Format; and index entries appearing in roman type indicate a page where the word, phrase, or BNF nonterminal was used in a heading, Function, Syntax Rule, Access Rule, General Rule, Leveling Rule, Table, or other descriptive text.

999 • 1

— T —

TMPRL-022 • 1
TMPRL-023 • 3

