X3H2-95-373
DBL:LHR-009
October, 1995

ISO

International Organization for Standardization

ANSI

American National Standards Institute

ANSI TC X3H2

ISO/IEC JTC 1/SC 21/WG 3

Database

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

Author: Jim Melton (Editor)

References:

1) ANSI X3H2-95-367/DBL:LHR-003, *(ISO Working Draft) Framework (SQL/Framework)*, October, 1995

2) ANSI X3H2-95-368/DBL:LHR-004, *(ISO-ANSI Working Draft) Database Language SQL*, October, 1995

3) ANSI X3H2-95-369/DBL:LHR-005, *(ISO Working Draft) Call-Level Interface (SQL/CLI)*, October, 1995

4) ANSI X3H2-95-370/DBL:LHR-006, *(ISO Working Draft) Persistent Stored Modules (SQL/PSM)*, October, 1995

5) ANSI X3H2-95-371/DBL:LHR-007, *(ISO Working Draft) Host Language Bindings (SQL/Bindings)*, October, 1995

6) ANSI X3H2-95-372/DBL:LHR-008, *(ISO Working Draft) Global Transaction Management (SQL/Transaction)*, October, 1995

7) ANSI X3H2-95-373/DBL:LHR-008, *(ISO Working Draft) Temporal (SQL/Temporal)*, October, 1995

8) X3H2-95-081, *Minutes from ANSI X3H2 meeting*, February, 1995

9) X3H2-95-209, *Minutes from ANSI X3H2 meeting*, May, 1995

10) X3H2-95-298, *Minutes from ANSI X3H2 meeting*, June, 1995

11) DBL:LHR-001, *Minutes from ISO DBL RG meeting*, July, 1995

12) X3H2-95-361, *Minutes from ANSI X3H2 meeting*, Sepptember, 1995

1 Editorial Notes

The attached SQL/Temporal base document incorporates the changes arising from the ANSI X3H2 meeting held May, 1995, in Oklahoma City, OK, the ANSI X3H2 meeting held June, 1995, in Charleston, WV, the ISO/IEC JTC1/SC21 SQL/CLI DIS Editing Meeting held July, 1995, in Tornonto, Ontario, Canada, the ISO/IEC JTC1/SC21 SQL/PSM CD Editing Meeting continuation held July, 1995, in Tornonto, Ontario, Canada, the ISO/IEC JTC1/SC21/WG3 DBL RG meeting held July, 1995, in Ottawa, Ontario, Canada, and the ANSI X3H2 meeting held August, 1995, in Milwaukee, WI.

The following conventions are used in the base document:

1) Words or short phrases that have been approved by ANSI but not by ISO are indicated by:
   ANSI sample phrase
   while words or short phrases approved by ISO and not by ANSI are indicated by:
   ISO example.
   Short differences between ANSI and ISO are indicated by:
   ANSI ANSI version.
   ISO ISO version.

2) Longer sequences of text that have been approved by ANSI, later changed by ISO, and not yet reconsidered by ANSI are distinguished by:

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

---

separating the ANSI-specific text from the preceding text by the convention you see here, and from the following text by the single heavy rule below. This separator is used to identify information that may change when ANSI has the opportunity to consider changes accepted by ISO that affect the area indicated.

Similarly, longer sequences approved by ISO, changed by ANSI, and not yet reconsidered by ISO are distinguished by:

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

---

separating the ISO-specific text from the preceding text by the convention you see here, and from the following text by the single heavy rule below as before. This separator is used to identify information that may change when ISO has the opportunity to consider changes accepted by ANSI that affect the area indicated.

It is the intent of these conventions to clearly highlight those portions of the SQL3 base document that are not yet approved by both bodies. One hopes that this highlighting will make it easier for the two bodies to move closer, so that a single standard can result from their efforts.

3) I have created a number of additional separators. While recognizing that the proliferation of these separators makes the document somewhat complex to examine, I believe it to be necessary in order that we may know the precise state of each part of the document. In particular, you may find in this document separators as follows:

ISO Only–SQL3

This separator indicates that ISO has approved the text following the separator for SQL3. ANSI has not approved the text at all.

ANSI Only–SQL3

This separator indicates that ANSI has approved the text following the separator for SQL3. ISO has not approved the text at all.

4) I am usually able to produce a document that has changebars to indicate changes between the present document and the preceding version. New (inserted) and changed material is marked with a thin vertical bar like that next to this paragraph.

- Sample deletion mark

  Additionally, places in the current document where material from the preceding version has been deleted are marked with a bullet like the one preceding this paragraph. The note beside the bullet, if any, indicates the number of "entities" that have been deleted. Those entities are typically sentences, although they are sometimes text processor commands.

In Clause 2, "Changes and Notes", I have made a number of notes related to the changes that I have made to Reference 1 to derive this document.

In Clause 3, "Possible problems with SQL/Temporal", I have listed a number of possible problems in SQL/Temporal. This list should be viewed as a sort of "work item list", helping identify areas that should (must?) be resolved before the document is ready for public review. This material is divided into genuine problems versus those problems that simply *should* be addressed or for which "wordsmithing" is the likely answer; the second category is not viewed as sufficiently urgent to cause the document to fail a ballot for public review or advancement. Finally, there is a section for "Language Opportunities" that lists those items in which some interest has been expressed, but that are not felt to be sufficiently important to justify delaying the standard in their absence.

In Clause 4, "SQL/Temporal Features Not Approved by Both ANSI and ISO", I have listed what I believe to be the differences between ANSI-approved SQL/Temporal features and ISO-approved features at the time of publication of these Editor's Notes.

In Clause 5, "Guidelines for writing "user-friendly" change proposals", I have outlined some suggested guidelines that I encourage you to use when writing change proposals. Following those guidelines will markedly help improve the quality of the document.

2 Changes and Notes

ISO DBL RG changes

In its July, 1995, meeting in Ottawa, Ontario, Canada, the ISO/IEC JTC1/SC21/WG3 Database Languages Rapporteur Group (DBL RG) Meeting considered all of the papers related to SQL/Foundation that were put before it.

X3H2 changes—Part 1

In its May, 1995, meeting in Oklahoma City, OK, ANSI Technical Committee X3H2 Meeting considered all of the papers related to SQL/Framwork that were put before it.

X3H2 changes—Part 2

In its June, 1995, meeting in Charleston, WV, ANSI Technical Committee X3H2 Meeting considered all of the papers related to SQL/Framwork that were put before it.

X3H2 changes—Part 3

In its September, 1995, meeting in Indianapolis, IN, ANSI Technical Committee X3H2 Meeting considered all of the papers related to SQL/Framwork that were put before it.

Other changes

In addition, the following changes appear in this document.

1) A number of minor editorial changes intended to clarify vague wording, incorrect grammar, inconsistent phraseology, and so forth.

3 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-001 Discussion of Paper X3H2-94-276/DBL:YOW-125 caused Paul Cotton to note the following Possible Problem:

> The base document is missing parts for dynamic SQL descriptors and using clause subclauses.

TMPRL-002 Discussion of Paper X3H2-94-329/DBL:YOW-155 noted the following Possible Problem:

> Since Subclause 4.2.1, "Type conversions and mixing of data types", in Part 2 of ISO/IEC 9075 specifies items of type period to be mutually comparable only if they have the same prcision, it is misleading not to make the corresponding statement about assignability in an analogous way. It is necessary to refer to Clause 8, "Data assignment rules", to see that assignment requires "the same period type", which is presumably intended to mean "the period type with the same percision".
>
> If this interpretation is correct, [Mike Sykes is] not in favor of it. In the comparison of values of data types that have different fractional precisions (i.e., exact numeric scale, timestamp and interval fractional seconds precision), SQL consistently provides for such precisions to be aligned.

TMPRL-003 Discussion of Paper X3H2-94-329/DBL:YOW-155 noted the following Possible Problem:

> [Mike Sykes] can find no specification of a less-than operation for values of data type period. This is surely necessary for ORDER BY. Admittedly, it can be argued that the ordering must be somewhat arbitrary, but even an ordering on the row value constructors (beginning timestamp, ending timestamp) would at least be better than nothing.

TMPRL-004 Discussion of Paper X3H2-94-329/DBL:YOW-155 noted the following Possible Problem:

In Subclause 5.2, "<literal>", General Rule 4), which enforces validation of periods, should be moved to Subclause 6.1, "<data type>" (where the corresponding rule for datetime is). It should also be amended to prohibit null delimiting timestamps.

TMPRL-005 Discussion of Paper X3H2-94-329/DBL:YOW-155 noted the following Possible Problem:

In Subclause 6.1, "<data type>", Syntax Rule 3) defines the length of a PERIOD as "43 positions...". [Mike Sykes believes] this to be inappropriate here (the same applies to the precedent that is being followed here).

See DBL:YOW-034 for a better suggestion, viz. set the descriptor field LENGTH to the number of characters sufficient to hold the longest result of CAST (whatever TO CHARACTER) for a value of the data type.

TMPRL-006 Discussion of Paper X3H2-94-329/DBL:YOW-155 noted the following Possible Problem:

In Subclause 6.2, "<set function specification>", PERIOD must be the only data type for which MIN or MAX may return a value that is not in the grouped table. This seems an undesirable departure from precedent.

TMPRL-007 Discussion of Paper X3H2-94-329/DBL:YOW-155 noted the following Possible Problem:

In Subclause 6.4, "<period value function>", General Rule 2)f) is redundant (because of the issue mentioned in Possible Problem **TMPRL-006**). Even if the redundancy is felt worthwhile (and [Mike Sykes doesn't] believe it is), then General Rule 2)f) should at least be promoted to a full General Rule, so as to apply also to PERIOD.

TMPRL-008 Discussion of Paper X3H2-94-329/DBL:YOW-155 noted the following Possible Problem:

In Subclause 6.5, "<cast specification>", the ability to cast from PERIOD to DATE or TIMESTAMP seems of small utility, especially in comparison with some of the features left out.

TMPRL-009 Discussion of Paper X3H2-94-329/DBL:YOW-155 noted the following Possible Problem:

[Mike Sykes] can discover no explicit way to obtain the value of the interval that represents the duration of a period. Nonetheless, the General Rules of Subclause 6.5, "<cast specification>", refer to "period of duration" rather than specifying an "ending timestamp". [Mike feels] this to be paricularly regrettable, since the user must know what (negative) power of 10 to add after subtracting BEGIN(P) from END(P).

TMPRL-010 Discussion of Paper X3H2-94-329/DBL:YOW-155 noted the following Possible Problem:

In Subclause 7.2, "<comparison predicate>", General Rule 1)a)iv), "$X - Y$" should presumably read "$X = Y$".

TMPRL-011 Discussion of Paper X3H2-94-329/DBL:YOW-155 noted the following Possible Problem:

In Subclause 7.3, "<overlaps predicate>", [Mike Sykes concurs] with the Editor's note and [is] concerned.

The operands of OVERLAPS are <row value expression>s. The data type of a <row value expression> is a row type, see for example DBL:YOW-004, Subclause 7.2, "<row value expression>", Syntax Rule 4). Yet, here, Syntax Rules 1) and 2) suggest that they may be PERIOD or TIMESTAMP. This comment applies also to the three following Subclauses.

[Mike] would expect the result to be true also in the case that B1 >= E2 AND B2 >= E2.

TMPRL-012 Discussion of Paper X3H2-94-329/DBL:YOW-155 noted the following Possible Problem:

In Subclause 7.6, "<meets predicate>", apart from the Editor's Note, [Mike Sykes] would expect the result to be true also in the case that B1 = E2 + 10^whatever.

Minor Problems and Wordsmithing Candidates:

Language Opportunities

4 SQL/Temporal Features Not Approved by Both ANSI and ISO

In this section, I list (in no particular priority) the differences I observe in ANSI- and ISO-approved SQL/Temporal features.

4.1 Standing differences without recent papers to reconsider

In this first list, I include differences that were caused by papers either not approved or not considered at the most recent meeting(s) of ANSI and/or ISO.

None.

4.2 Standing differences with recent papers

In this second list, I include those differences that remain outstanding even though papers were considered at the most recent meeting(s) of ANSI and/or ISO, but that were not accepted or that did not completely resolve the differences.

None.

4.3 Recent differences

In this next list, I include those differences resulting from the most recent meeting(s) of ANSI and/or ISO, that the other body has not yet had the opportunity to consider.

None.

5 Guidelines for writing "user-friendly" change proposals

This chapter of the Editor's Notes offers guidelines to the style of the content of the SQL document and all its Parts, as well as some guidelines for writing change proposals.

The Editor reserves the right to reject any change proposals that do not follow these guidelines, as appropriate.

5.1 Style of content of SQL document

5.1.1 General

5.1.1.1 Language

Since the SQL standard will be translated into other languages, and must in any case be read and understood by many whose first language is not English, simplicity of language is much more important than literary elegance.

Normal rules of good style apply; if in doubt, there are several excellent references to writing style that may be consulted.

The following specific points are noted:

— Conditions: The standard form used is "If *some condition* is satisfied, then *something happens*." This is strongly preferred to "*X* happens if *Y* is true." The "otherwise" case always deserves consideration, though it doesn't always need to be stated (particularly when the otherwise case involves no action or behavior).

— Number: It is clearer to stick to the singular and say, for example, "every *X* is destroyed" than to say "all *x*s are destroyed". Where "each" is acceptable, it is even better.

— Avoid the use of the word "any" as far as practicable. In ordinary English, it sometimes means "some" (as in "have you any shares in Company X?"), and sometimes means "every" (as in "you can call any day, any time").

  In particular, avoid constructions like "Let *CD* be any collation descriptor that includes . . . ". Use precise quantification, such as "For every collation descriptor *CD* that includes . . . ", and note that it is not necessary to add "if any"—even if there is no such collation descriptor, the rule still works fine.

— "which" versus "that": See X3H2-88-036 (copies available upon request). "That" should be used much more often than many people do.

5.1.1.2 Vocabulary

English is well known for having many synonyms (change, alter, amend), partially overlapping synonyms (describe and define, change and improve), and near-synonyms with subtle differences (edible and eatable), or even ignored distinctions (expect and anticipate). It may be assumed that any reader has a dictionary available, but where a word can be used in more than one sense, the correct interpretation should not depend on context any more than is absolutely necessary. If a word can have the same meaning or translation into another language as some other word used in SQL3 and no distinction is intended, then that other word should be used in preference.

Particular words used in specific senses in SQL3 include:

— **Definition**: Either a BNF non-terminal that causes the creation of something, or a definition of the meaning of a term.

— **Descriptor**: A persistent, formal description of an SQL object. See Subclause 3.3.5, "Descriptors", in Part 1.

— **Specify**: To state explicitly, for example "If GLOBAL is specified, . . . " means "If the word 'GLOBAL' actually appears, . . . ". Note that " . . . is specified . . . " appears more frequently than " . . . was specified . . . ".

— **Case**: is favored. Nested cases are preferred to complex ones. Only the first rule is applied for which the condition is satisfied, so the *order* of the subrules is important. A Case construct normally ends with an "Otherwise" subrule, but this is not necessarily required.

Other terms worth noting are:

— **Database**: Note: this is one word. A collection of SQL-data. The term is to be avoided, because it raises the question of whether databases can overlap, be nested, are each described by a catalog or cluster of catalogs, *etc.* Usually, the term "SQL-data" is sufficient.

— **Description**: An informal description; not used in a technical sense. Note that the object that serves as a persistent description of an SQL object is known as a *descriptor*.

5.1.1.3 Quantification

When universally quantifying, prefer the singular "every" to the plural "all"—it nearly always works better. Although mathematicians often say "for all *x*", it is not really good grammar (because the upside-down A (∀) of predicate calculus is considered to stand for "all", the first letter of "every" having been taken by the backwards E (∃) that stands for "exists").

When existentially quantifying, use "some", not only in preference to "any", but also sometimes in preference to the indefinite article "a" or "an", which can be as ambiguous as "any". However, this suggestion can lead to "overkill" text, so we further suggest that the indefinite article is appropriate in cases where it is clear that there can be no more than one occurrence of the thing in question, as in : "For every collation descriptor that includes a <translation collation> . . . " (a collation descriptor includes at most one <translation collation>, so the word "some" instead of "a" could even be misleading here).

5.1.1.4 Quotation marks

Single <quote>s enclose <character string literal>s; <double quote>s enclose <delimited identifier>s. In text, double quotes are normally used to surround quoted material.

5.1.1.5 Typography

— <key word>s (only) are in <simple Latin upper case letter>s.

— Truth values (as opposed to Boolean values) are lower-case, italicized, and underlined: *true* , *false* , and *unknown* .

— With the exception of certain index entries and certain table headings, the only bolding in the document is of:

- Ada keywords such as **package**, because the Ada convention is followed;
- The terms being defined in Subclause 3.1, "Definitions"; and
- Names of pseudo-procedures (visible now only in the definition of significant DB_changes, where the procedure **proc_VC** is defined).

— Clauses ("chapters" to some) are the major divisions in the document; Subclauses are all the lesser divisions. The names of Clauses and Subclauses are all spelled with initial capital letters; subsequent words in those names are spelled with initial lower-case letters—except, of course, for such words as "SQL".

5.1.2 SQL terminology

5.1.2.1 Terms for SQL objects

Terms referring to SQL objects must be carefully chosen, and used not only correctly but wherever possible.

Such terms are frequently prefixed with "SQL-" in order to distinguish them from similar terms used with different meanings in related contexts. For example: SQL-transaction, SQL-session, SQL-client. Note that SQL-schema is, strictly, the correct term, because it has a different meaning from "schema" in the Reference Model of Data Management, but "schema" is used throughout the SQL document because it is always the SQL sense that is intended.

Ambiguities such as "object identifier" should be avoided.

5.1.2.2 BNF non-terminals

The names of BNF non-terminals do not often contain hyphens; where they do, it is normally because the analogous English phrase would contain hyphens ("form-of-use", "implementation-defined"). They sometimes contain colons to distinguish closely related constructs (for example, <delete statement: positioned> and <delete statement: searched>).

Care should be taken not to introduce over-general names. For example, <SQL statement> is unfortunate in that its production rule does not allow any SQL-statement.

Every new <key word> must be specified as a <reserved word> or <non-reserved word>, as appropriate.

Precision in referring to BNF non-terminals is important. For example: there is no such thing as a <statement>.

Do not fall into the trap of assuming that the name of a BNF non-terminal means the same thing as the words contained within the angle brackets. When writing a rule about <select list>, do not then start using the phrase "select list" as though it has some meaning. BNF non-terminals are nothing more than distinguished character strings whose spelling is irrelevant except for consistent use. If "<select list>" were replaced "automatically" with "<SL>", one would not feel comfortable discussing the "SL" as a normal English phrase.

5.1.2.3 BNF non-terminal or name of SQL object?

BNF non-terminals do not denote persistent objects. In particular, it is incorrect to say "If an <assertion definition> exists . . . "; an <assertion definition> is source code, which, once processed, ceases to exist as far as the SQL implementation is concerned. The result of processing it is an *assertion*, represented by an *assertion descriptor* in some schema.

data type or <data type>? "data type" is the correct term, unless it is specifically intended to refer to the BNF nonterminal "<data type>" (probably appropriate only when discussing the syntax).

Where reference is intended to a generic data type, such a character string type, then that is the preferred term.. SQL, at the time of writing, uses "character data type", "<character string type>", "data type CHARACTER", and several other phrases almost synonymously.

"<preparable statement>" is a BNF non-terminal; "prepared statement" is not.

5.1.2.4 Notes on specific SQL terms

"contain" is used for syntactic containment, *i.e.*, one BNF non-terminal is said to contain others. "data type" is two words. "datetime" is one word. "include" is used for specifying that one descriptor is included in others.

5.1.2.5 Locally-defined terms and symbols

Diacritical or other marks, such as prime, are to be avoided for typographical reasons.

Other considerations involving terms and symbols:

— Symbolic names are italicized and composed of <simple Latin upper case letter>s and <digit>s. Examples: *T*, *C1*, and *SLCC*.

— Indexes or subscripts are in lower-case italic letters and digits. Examples: "the *i*-th column", "C_k", or "at least *j* values".

— Special symbols for use as "range variables" (like the "CD" earlier) should not be introduced unnecessarily. For example:

> Every aardvark-reference *A* that contains . . . is removed from every zoological paper that includes *A*.

is preferable to:

> For every zoological paper *ZP*, for every aardvark-reference *A* that contains . . . and is included in *ZP*, *A* is removed from *ZP*.

Special symbols are necessary to avoid clumsiness and to ensure precision in many of the more complicated rules.

These special symbols' scope is the Subclause in which they are defined. While they may be defined early in a Subclause (for example, in the Syntax Rules) and used much later (for example, in the General Rules), it is preferable to define them in the section in which they are used within a Subclause.

5.1.2.6 Abbreviations

Abbreviations such as "auth-id" are not used in the SQL standard, however widely they may be generally used and understood. It follows that they are not acceptable in text proposed for insertion in the document.

5.1.3 The content of the SQL standard

5.1.3.1 Definitions

This section should conform to the (normative!) Annex B.1 of the ISO Directives, Part 3.

A term should be defined in this section if and only if:

— It is used through the Standard, rather than in any particular context; and

— Either:

- It is a term not widely used or that has been invented for the Standard; or
- It has more than one meaning and a precise meaning is consistently intended in the Standard.

A term should not be included simply for tutorial purposes (the Standard is already large enough).

5.1.3.2 Concepts

Although Clause 4, "Concepts", was originally introduced purely for the purpose of explanation, it now contains much material that is definitive. There are no guidelines to what should or should not be there. Thus Subclause 4.2.3, "Rules determining collating sequence usage", are classed as concepts, while tables of valid casts and valid values of datetime data types are not.

5.1.3.3 Modularity

Except perhaps where the result would be a ridiculously short Subclause, the specification of a BNF non-terminal used in more than one place should be in a separate Subclause from every one of its uses. Otherwise, there is a serious risk of unintended effects resulting from changing the Rules in one place without realizing that they should remain unchanged for the use elsewhere.

In particular, the construct "Every General Rule except General Rule 1 of Subclause x.y applies here with 'something' replaced by 'something else'" is to be avoided at all costs.

5.1.3.4 Format rules

The syntax of the version of BNF used is defined in Subclause 3.2, "Notation".

Every SQL character that is not a letter or digit has a name, defined in Subclause 5.1, "<SQL terminal character>". Except in those BNF productions that define names for them, such characters never appear standing for themselves, but are always represented by their names. Thus, except where they are specified as <left bracket>, *etc.*, the characters "[", "]", "{", "}", " | ", and " . . . " in BNF productions always serve as "punctuation".

Therefore, the production:

```
<x comma list> ::= ( <x> [ , <x> ]... )
```

should always be written as:

```
<x comma list> ::= <left paren> <x> [ <comma> <x> ]... <right paren>
```

5.1.3.5 Syntax and Access Rules, and General Rules

The differences between Syntax and Access Rules on the one hand and General Rules on the other are that Syntax and Access Rules specify requirements for a program to conform to (some level of) the SQL standard, while General Rules specify what a standard-conforming SQL implementation is required to do when it process such a standard-conforming program.

Where there are no Rules of any kind, it is usual to say "None.", to exclude the possibility that they have not been thought about, or have been lost.

5.1.3.6 Syntax Rules

The correct form is "This condition shall be satisfied . . . " or "If *X* is specified, then *Y* shall be specified".

Syntax Rules *do not* deal with the privileges required to accomplish some action.

5.1.3.7 Access Rules

The correct form is as for Syntax Rules.

Access Rules *do* deal with the privileges required to accomplish some action.

5.1.3.8 General Rules

General Rules specify the effect of processing SQL source code that satisfies the relevant Format, Syntax Rules, and Access Rules. The preferred form is present indicative passive, *e.g.*, "A descriptor is created . . . ".

By the time the Syntax and Access Rules have been processed, certain implications have become explicit. Thus, if the optional <schema name> of a <table name> is omitted, it is deduced, and by the time General Rule 1) is encountered, it can be assumed that <schema name> is known. It is not normally necessary or desirable to mention that it is "explicit or implicit".

5.1.3.9 Exceptions

There are two categories of end-of-statement behavior:

— " . . . an exception condition is raised: *some condition*"; and

— " . . . a completion condition is raised: *limited alternatives*". The only acceptable alternatives for the completion conditions are *successful completion*, *warning*, and *no data*. Each of these may have no subcode or a specific subcode.

The exception class and subclass phrases are separated by a dash "—" and are always both in italics.

5.1.3.10 Leveling Rules

The correct form is as for Syntax Rules. A Rule that illustrates a point by the use of an SQL expression either shows the SQL expression on a separate line:

Let *B* be an exact numeric result of the operation:

CAST (CAST (*Y* AS *Q*) AS *E2*)

or encloses the expression in double-quotes:

"*R* is NULL" is true if and only if . . .

5.2 Writing change proposals

5.2.1 Discussion

It is helpful to the reader if the discussion part makes quite clear why the proposal is being made. It also helps if any approaches that were considered and rejected are also mentioned, together with the reasons for their rejection.

Proposal writers should strive to describe *all* changes and their implications in the discussion part. Readers should have have to slog through the detailed language changes to understand the consequences of the change.

5.2.2 Change proposals

A) Every separate item of the proposal should be numbered.

B) When referring to a Subclause of the document, always specify both Subclause number and the name of the Subclause (for example, Subclause 6.1, "<data type>").

C) When inserting, deleting, or replacing text or Rules in the document, specify both the reference, *e.g.*, Syntax Rule 11)c)iv)2)B), and sufficient context to identify the specific paragraph, sentence, or Rule to be changed.

   For example, to replace Subclause x.y, Syntax Rule 11)c)iv)2)B), each of Syntax Rules 11), 11)c), 11)c)iv), and 11)c)iv)2) should be unambiguously defined. Where, as for example in Subclause 12.9, "<using clause>", several Rules start off very similarly, say something like "Replace Syntax Rule 11)c)iv)2)B) (the Rule that deals with the *xyz* in an *rst*) with...".

D) Where only a few words of existing text are to be changed, it is clearer both to the reviewer and the Editor to flag differences in some way, *e.g.*, by striking out deleted text and underlining new. Whatever convention is used should be explained.

5.2.3 Check list

All changes to the document must be specified explicitly. It is unfair to the Editor to expect him to make changes left implicit.

The following list is provided in the hope of reducing the number of incomplete change proposals:

- Leveling Rules and Appendix A, "Leveling the SQL Language",
- Appendix B, "Implementation-defined elements" or Appendix C, "Implementation-dependent elements",
- Appendix E, "Incompatibilities with X3.135-1992 and ISO/IEC 9075:1992", especially for new <reserved word>s,
- Appendix E, "Incompatibilities with X3.135-1992 and ISO/IEC 9075:1992", for new class or subclass values—also please remind the Editor to index the new SQLSTATE values; also update the table of Ada package values in SQL/Bindings, Subclause 9.2, "<routine>".
- The Dynamic SQL Descriptor Area
- Information and Definition Schemas.

A proposal that solves a Possible Problem, Minor Problem, or Opportunity, whether intentionally or not, should say so prominently, so that the Editor's Notes may be kept up-to-date.

A proposal that introduces a Possible Problem, for example because it is admittedly incomplete, should specify an appropriate addition to the Possible Problems list. The same applies to language opportunities.

A proposal to insert lengthy pieces of text into the document should, either before or as soon as practicable after it has been adopted, be provided to the Editor in machine-readable form, either on diskette or by electronic mail (see Clause 6, "Machine readable change proposals" for more information).

It is accepted that this machine-readable version will be in "plain text" form and without many formatting attributes (*e.g.*, bolding, underlining, font changes), but it is nevertheless of great help to the Editor.

Because of the increasing size and partitioning of the SQL3 document and the great complexity of dealing with proposals that make substantial changes to the document's structure and content, I reserve the right to decline instructions to change the document when those instructions are not complete and that do not generally follow these guidelines.

Specifically, if a change proposal is accepted that fails to cite the number and title of the document being changed by the proposal, the proper Clause and/or Subclause numbers and titles affected by the proposal, and the Rule numbers and partial text of Rules that are changed by the proposal, then I will not make any changes in the next edition of the base document, but will bring the paper back to the Committee for revision and completion. Thanks!

6 Machine readable change proposals

I would like to thank those ISO and ANSI participants who have provided me with copies of change proposals in machine-readable form. This practice reduces the editorial workload considerably and makes it much easier to ensure accurate reflection of the change proposals in the base document. I continue to urge those participants who can supply diskette copies of lengthier proposals (i.e., a page or more of new text) to bring those diskettes to meetings, in order to avoid mail delays. I will of course either return the diskettes or replace them with blank diskettes on request.

I prefer 3.5 inch DS/HD soft-sectored diskettes, with DOS files (clearly labeled in any case!). I may be able to find ways to copy other types of diskette and file, but would prefer to avoid the hassles if possible. Please include both unformatted and formatted versions of each file, if possible. Fully-justified text (with the extra blanks inserted for alignment purposes) do not appear to be a problem for my document processor.

It is probably preferable for you to electronically mail me your change proposals. My "generic" Internet mail address is:

jim.melton@sybase.com

Thanks!

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.

ISO Working Draft

Database Language SQL — Part 7: Temporal (SQL/Temporal)

«Part 7»

October 1995

Contents

Page

TABLES

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.

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

International Standard ISO/IEC 9075, Part 7, was prepared by Joint Technical Committee ISO/IEC JTC 1, Information technology.

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

— Part 1: Framework (SQL/Framework)

— Part 2: Foundation (SQL/Foundation)

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

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

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

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

— Part 6: XA Specialization (SQL/Transaction)

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

— Part 6: Global Transaction Management (SQL/Transaction)

— Part 7: Temporal (SQL/Temporal)

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 the Information Standard 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, “Predicates”, defines a number of predicates used in the manipulation of temporal support.

8) Clause 8, “Data assignment rules”, specifies the rules for assignments that retrieve data from or store data into the database, and formation fules for set operations.

9) Clause 9, “Schema definition and manipulation”, defines facilities for creating and managing a schema.

10) Clause 10, “Information Schema and Definition Schema”, defines viewed tables that contain schema information.

11) Clause 11, “Status codes”, defines SQLSTATE values related to temporal support.

12) Clause 12, “Conformance”, defines the criteria for conformance to this Part of this International Standard.

13) Annex A, “Implementation-defined elements”, is an informative Annex. It lists those features for which the body of this International Standard states that the syntax or meaning or effect on the database is partly or wholly implementation-defined, and describes the defining information that an implementor shall provide in each case.

14) Annex B, “Implementation-dependent elements”, is an informative Annex. It lists those features for which the body of this International Standard states explicitly that the meaning or effect on the database is implementation-dependent.

15) Annex C, “Deprecated features”, is an informative Annex. It lists features that the responsible Techical Committee intends will not appear in a future revised version of this International Standard.

16) Annex D, "Incompatibilities with ISO/IEC 9075:1992", 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 predefined data type named PERIOD; and

— the specification of scalar expressions and predicates used to manipulate values of the PERIOD data type.

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

2 Normative references

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

ANSI Only–SQL3

— ANSI X3.9-1978, *American National Standard Programming Language FORTRAN*.

— ANSI X3.23-1985, *American National Standard for Information Systems—Programming Language—COBOL*.

— ANSI X3.53-1976, *American National Standard Programming Language PL/I*.

— ANSI X3.135-1992, *American National Standard for Information Systems — Database Language — SQL*.

— ANSI X3.159-1989, *American National Standard for Information Systems—Programming Language—C*.

— ANSI X3.198-1991, *American National Standard for Information Systems—Programming Language—Fortran*.

NOTE 2 – ANSI X3.198-1991 introduces no incompatibilities with ANSI X3.9-1978 that affect the binding between Fortran and SQL; therefore, wherever "Fortran" is specified in this International Standard, ANSI X3.198-1991 is implicit.

— ANSI/MDC X11.1-1990, *American National Standard for Information Systems—Programming Language—MUMPS*.

— ANSI/IEEE 770/X3.97-1983, *American National Standard for Information Systems—Programming Language—Pascal*.

— ANSI/IEEE 770/X3.160-1989, *American National Standard for Information Systems—Programming Language—Extended Pascal*.

— ANSI/MIL-STD-1815A-1983, *American National Standard for Information Systems—Reference Manual for the Ada® Programming Language*.

ISO Only–SQL3

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

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

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

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

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

— ISO/IEC 9075-1:199*x*, *Information Technology — Database Languages — Framework for SQL*.

— ISO/IEC 9075-2:199*x*, *Information Technology — Database Languages — SQL Foundation*.

— ISO/IEC 9075-3:199*x*, *Information Technology — Database Languages — SQL Call-Level Interface*.

— ISO/IEC 9075-4:199*x*, *Information Technology — Database Languages — SQL Persistent Stored Modules*.

— ISO/IEC 9075-5:199*x*, *Information Technology — Database Languages — SQL Host Language Bindings*.

— ISO/IEC 9075-6:199*x*, *Information Technology — Database Languages — SQL Global Transaction Support*.

— ISO/IEC 9899:1990, *Information technology — Programming languages — C*.

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

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

ISO Only–SQL2

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

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

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

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

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

— ISO/IEC 9075:1992, *Information Technology — Database Languages — SQL*.

— ISO/IEC 9899:1990, *Information technology — Programming languages — C*.

— 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) **period**: a set of contiguous granules of a specified precision.

b) **granule (of a specified precision *P*)**: one particular 10^{-P} second.

c) **beginning delimiting timestamp**: the first granule of a period.

d) **ending delimiting timestamp**: the last granule of a period.

More to be supplied as required.

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 convensions 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
SQL2 ISO/IEC 9075:1992.
SQL3 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
SQL2 ISO/IEC 9075:1992.
SQL3 Part 2 of this International Standard.

3.3.1 Subclause naming

Clauses and Subclauses in this Part of this International Standard that have names identical to Clauses or Subclauses in
SQL2 ISO/IEC 9075:1992
SQL3 Part 2 of this International Standard
supplement the Clause or Subclause, respectively, in
SQL2 ISO/IEC 9075:1992,
SQL3 Part 2 of this International Standard,
typically by replacing Format items or Rules or by providing new Format items or Rules.

Clauses and Subclauses in this Part of this International Standard that have names that are not identical to Clauses or Subclauses in
SQL2 ISO/IEC 9075:1992
SQL3 Part 2 of this International Standard
provide language specification particular to this Part of this International Standard.

3.4 Object identifier for Database Language SQL

NOTE 3 – 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 Be Supplied.

4.2 Period data type

A period data type is described by a period data type descriptor. A period data type descriptor contains:

— the name of the period data type (PERIOD or PERIOD WITH TIME ZONE); and

— the value of the <time fractional seconds precision>.

Every period data type has an implied length in positions. Let *V* denote a value in some period data type *PT*. The length in positions of *PT* is constant for all *V*. The length in positions is the number of characters from the character set SQL_TEXT that it would take to represent any value in a given period data type.

Table 1, "Valid operators involving periods and intervals", specifies the results of arithmetic expressions involving period and interval operands.

Table 1—Valid operators involving periods and intervals

Operand 1	Operator	Operand 2	Result Type
Period	+	Interval	Period
Period	–	Interval	Period
Interval	+	Period	Period

Arithmetic operators involving a period and an interval preserve the time zone of the period operand. If the period operand does not include a time zone part, then the local time zone is effectively used.

A period is a set of contiguous granules of a specified precision. For a period data type with precision *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.

4.2.1 Type conversions and mixing of data types

Values of type period are mutually assignable.

Items of type period are mutually comparable only if they have the same precision. Period *X precedes* period *Y* if the last granule of *X* is less than the first granule of *Y*. Period *X contains* period *Y* if the first granule of *X* is less than or equal to the first granule of *Y* and the last granule of *X* is greater than or equal to the last granule of *Y*. Period *X meets* period *Y* if the last granule of *X* is adjacent to the first granule of *Y*. Period *X overlaps* period *Y* if the first granule of *X* is less than or equal to the last granule of *Y* and the first granule of *Y* is less than or equal to the first granule of *X*.

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 Part 2 of this International Standard
       | <period string>

<reserved word> ::=
         !! All alternatives from Part 2 of this International Standard

       | CONTAINS

       | MEETS

       | PERIOD | PRECEDES
```

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 Part 2 of this International Standard
       | <period literal>

<period literal> ::= PERIOD <period string>

<period string> ::=
       <quote> <left bracket>
         <beginning timestamp> <space> <minus sign> <space> <ending timestamp>
           [ <time zone interval> ]
       <right bracket> <quote>

<beginning timestamp> ::=
       <date value> <space> <time value>

<ending timestamp> ::=
       <date value> <space> <time value>
```

Syntax Rules

1) The data type of a <period literal> that does not specify <time zone interval> is PERIOD(*P*), where *P* is:

   a) If a <seconds fraction> *SF* is specified in the <time value> simply contained in <beginning timestamp>, then the number of digits in *SF*.

   b) Otherwise, 0.

2) The data type of a <period literal> that specifies <time zone interval> is PERIOD(*P*) WITH TIME ZONE, where *P* is:

   a) If a <seconds fraction> *SF* is specified in the <time value> simply contained in <beginning timestamp>, then the number of digits in *SF*.

   b) Otherwise, 0.

3) The <time value> simply contained in <ending timestamp> shall have the same number of digits in its <seconds fraction> as the <time value> simply contained in <beginning timestamp>.

Access Rules:

No additional Access Rules.

General Rules

1) The beginning and ending delimiting timestamps of a period shall have the same precision as the period.

   NOTE 4 – Beginning and ending delimiting timestamps are defined in Subclause 4.2, "Period data type".

2) Let *A* and *B* be valid <date value>:<time value> pairs, representing timestamps *C* and *D*. If the <period string> of a <period literal> is identical to

   '[*A* - *B*]'

   or

   '[*A* - *B* <timezone interval>]' ,

   then the value of the <period literal> is the period from *C* to *D*, inclusive.

3) If <time zone interval> is specified, then the value of <period literal> represents a period in the specified time zone; otherwise, the value of <period literal> represents a period in the current default time zone of the SQL-session.

4) The valid values for fields of the delimiting timestamps in a period data type are constrained identically to those in the TIMESTAMP data type. If any specification or operation attempts to cause an item of type period to take a value where the beginning delimiting timestamp is greater than the ending delimiting timestamp, then an exception condition is raised: *data exception — invalid period format*.

6 Scalar expressions

6.1 <data type>

Function

Specify a data type.

Format

```
<predefined type> ::=
         !! All alternatives from Part 2 of this International Standard
       | <period data type>

<period data type> ::=
       PERIOD [ <left paren> <period precision> <right paren> ] [ WITH TIME ZONE ]

<period precision> ::=
       <time fractional seconds precision>
```

Syntax Rules

1) If <period precision> is not specified, then 6 is implicit.

2) PERIOD specifies the data type period.

3) The length of a PERIOD is 43 positions plus the <time fractional seconds precision> times 2 plus 2 positions if the <time fractional seconds precision> is greater than 0. The length of a PERIOD WITH TIME ZONE is 49 positions plus the <time fractional seconds precision> times 2 plus 2 positions if the <time fractional seconds precision> is greater than 0.

Access Rules

No additional Access Rules.

General Rules

1) For a <period data type>, a <time fractional seconds precision> that is an explicit or implicit <period precision> defines the number of decimal digits following the decimal point in the SECOND <datetime field> of the delimiting timestamps of the period.

2) If WITH TIME ZONE is not specified, then the time zone displacement of the period data type is effectively the current default time zone displacement of the SQL-session.

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

2) Let *T* be the argument or argument source of a <set function specification>.

3) If SUM is specified, then *DT* shall not be a period data type.

4) If AVG is specified, then *DT* shall not be a period data type.

Access Rules

No additional Access Rules.

General Rules

1) Case:

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

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

2) If MIN is specified and *DT* is a period data type, then:

   a) Let *TS* be a timestamp that is equal to the beginning delimiting timestamp of one of the periods in *TX*, such that there exists no beginning delimiting timestamp *TS2* of any period in *TX* and *TS2* precedes *TS*.

   b) Let *TE* be a timestamp that is equal to the ending delimiting timestamp of one of the periods in *TX*, such that there exists no ending delimiting timestamp *TE2* of any period in *TX* and *TE2* precedes *TE*.

   c) The result is the period with beginning and ending delimiting timestamps *TS* and *TE*, respectively.

3) If MAX is specified and DT is a period data type, then:

   a) Let *TS* be a timestamp that is equal to the beginning delimiting timestamp of one of the periods in *TX*, such that there exists no beginning delimiting timestamp *TS2* of any period in *TX* and *TS* precedes *TS2*.

b) Let *TE* be a timestamp that is equal to the ending delimiting timestamp of one of the periods in *TX*, such that there exists no ending delimiting timestamp *TE2* of any period in *TX* and *TE* precedes *TE2*.

c) The result is the period with beginning and ending delimiting timestamps *TS* and *TE*, respectively.

6.3 <datetime value function>

Function

Specify a function yielding a value of type datetime.

Format

```
<datetime value function> ::=
         !! All alternatives from Part 2 of this International Standard
       | BEGIN <left paren> <period value expression> <right paren>
       | END <left paren> <period value expression> <right paren>
```

Syntax Rules

1) The data type of a <datetime value function> simply containing a <period value expression> of data type PERIOD is TIMESTAMP. The data type of a <datetime value function> simply containing a <period value expression> of data type PERIOD WITH TIME ZONE is TIMESTAMP WITH TIME ZONE.

Access Rules

No additional Access Rules.

General Rules

1) The <datetime value function>s BEGIN and END respectively return the beginning and ending delimiting timestamps of the <period value expression> with a precision identical to the precision of the <period value expression>.

6.4 <period value function>

Function

Specify a function yielding a value of type period.

Format

```
<period value function> ::=
         PERIOD
           <left paren>
             <datetime value expression 1> <comma> <datetime value expression 2>
           <right paren>
       | INTERSECT
           <left paren>
             <period value expression 1> <comma> <period value expression 2>
           <right paren>

<datetime value expression 1> ::= <datetime value expression>

<datetime value expression 2> ::= <datetime value expression>

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

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

Syntax Rules

1) If PERIOD is specified, then

   Case:

   a) If <datetime value expression 1> is of type DATE, then <datetime value expression 2> shall be of type DATE.

   b) Otherwise <datetime value expression 1> and <datetime value epxression 2> shall be of type TIMESTAMP with identical precisions.

2) If INTERSECT is specified, then <period value expression 1> and <period value expression 2> shall have identical precisions.

3) The data type of a <period value function> is period, with a precision of:

   Case:

   a) If INTERSECT is specified, then the precision of <period value expression 1>.

   b) If <datetime value expression> is of data type DATE, then 0.

   c) If <datetime value expression 1> is of data type TIMESTAMP, then the precision of <datetime value expression 1>.

Access Rules

None.

General Rules

1) If PERIOD is specified and the <datetime value expression>s are of data type DATE, then let *FD* be <datetime value expression 1> and let *ED* be <datetime value expression 2>. Let *FT* be

   CAST (*FD* AS TIMESTAMP)

   and let *ET* be

   CAST (*ED* AS TIMESTAMP)

   Let *BL* be a <beginning timestamp> whose value is *BT* and let *EL* be an <ending timestamp> whose value is *ET*. The value of the <period value function> is the value of:

   PERIOD '[*BL* - *EL*]'

2) If INTERSECT is specified, then:

   a) Let *FP* be the value of <period value expression 1>.

   b) Let *SP* be the value of <period value expression 2>.

   c) Let *BT1* be the beginning delimiting timestamp of *FP*. Let *BT2* be the beginning delimiting timestamp of *SP*. If *BT1* > *BT2*, then let *BT* be *BT1*; otherwise, let *BT* be *BT2*.

   d) Let *ET1* be the ending delimiting timestamp of *FP*. Let *ET2* be the ending delimiting timestamp of *SP*. If *ET1* < *ET2*, then let *ET* be *ET1*; otherwise, let *ET* be *ET2*.

   e) Let *BL* be a <beginning timestamp> whose value is *BT* and let *EL* be an <ending timestamp> whose value is *ET*. The value of the <period value function> is the value of:

      PERIOD '[*BL* - *EL*]'

   f) If any specification of operation attempts to cause an item of type period to take a value whose beginning delimiting timestamp is greater than the ending delimiting timestamp, then an exception condition is raised: *data exception — invalid period format*.

6.5 <cast specification>

Function

Specify a data conversion.

Format

No additional Format items.

Syntax Rules

1) Let *TD* be the specified <data type>.

2) Let *SD* be the underlying data type of the <value expression>.

3) 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> *SD* of <value expression>	<data type> of *TD*																	
	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) If *TD* is the period data type, then

   Case:

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

      ```
      TRIM ( BOTH ' ' FROM SV )
      ```

      Case:

      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 let *TV* be that value.

      ii) Otherwise, an exception condition is raised: *data exception — invalid character value for cast*.

   b) If *SD* is the datetime data type DATE, then the conversion results in a period of duration one day with a beginning delimiting timestamp of *SV*, with the <datetime fields>s hour, minute, and second set to 0.

   c) If *SD* is the datetime date type TIMESTAMP, then the conversion results in a period of duration one granule at the precision of *SD* with a beginning delimiting timestamp of *SV*.

   d) If *SD* is the period data type, then *TV* is *SV*.

2) If *SV* is the PERIOD data type, then:

   Case:

   a) If *TD* is the data type DATE, then *SV* shall be of duration one day, and the conversion results in a DATE with <datetime fields>s year, month, and day equal to that of the beginning delimiting timestamp of *SV*.

   b) If *TD* is the data type TIMESTAMP, then *SV* shall be of duration one granule at the precision of *SD*, and the conversion results in a TIMESTAMP equal to that of the beginning delimiting timestamp of *SV*.

6.6 <value expression>

Function

Specify a value.

Format

```
<value expression> ::=
         !! All alternatives from Part 2 of this International Standard
       | <period value expression>
```

Syntax Rules

1) The data type of <period value expression> shall be a period data type.

Access Rules

No additional Access Rules.

General Rules

1) No additional General Rules.

6.7 <period value expression>

Function

Specify a period value.

Format

```
<period value expression> ::=
         <period term>
       | <interval value expression> <plus sign> <period value expression>
       | <period value expression> { <plus sign> | <minus sign> } <interval value expression>


<period term> ::=
       <period factor>

<period factor> ::=
       <period primary> [ <time zone> ]

<period primary> ::=
         <period literal>
       | <column reference>
       | <scalar subquery>
       | <case expression>
       | <period value function>
       | <cast specification>
```

Syntax Rules

1) The data type of a <period value expression> is period, with a precision of the <period value expression> that it simply contains.

2) If <time zone> is not specified, then "AT LOCAL" is implicit.

Access Rules

None.

General Rules

1) If the result of any <period primary> or <interval value expression> contained in a <period value expression> is
   SQL2 the
   SQL3 the
   SQL4 a
   null value, then the result of the <period value expression> is the
   SQL4 general
   null value.

2) If a <period value expression> immediately contains <plus sign> or <minus sign>, then the result is effectively evaluated as follows:

   a) Let *ST* and *ET* be the beginning and ending delimiting timestamps, respectively, of the <period value expression>.

   b) Let *ST2* be the result of performing the operation on *ST* and the value of <interval value expression>, as described in Subclause 6.20, "<datetime value expression>", in Part 2 of this International Standard.

   c) Let *ET2* be the result of performing the operation on *ET* and the value of <interval value expression>.

   d) The result will be the value computed by:

      ```
      PERIOD ( ST2, ET2 )
      ```

3) If <time zone> is specified or implied, then:

   a) If LOCAL is specified, then let *TZ* be the current default time zone displacement of the SQL session. Otherwise, let *TZ* be the value of the <simple value specification> simply contained in the <time zone>.

   b) If the value of the <interval value expression> immediately contained in <time zone specifier> is less than INTERVAL -'12:59' or greater than INTERVAL +'13:00', then an exception condition is raised: *data exception — invalid time zone displacement value.*

   c) Let *DV* be the value of the <period primary> directly contained in the <period value expression> expressed as a period normalized to *UTC*.

   d) The value of the <period value expression> is calculated as:

      ```
      DV - TZ
      ```

7 Predicates

7.1 <predicate>

Function

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

Format

```
<predicate> ::=
        !! All alternatives from Part 2 of this International Standard
      | <precedes predicate>
      | <meets predicate>
      | <contains predicate>
```

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

1) The result of a <predicate> is a truth value derived according to the General Rules of Subclause 7.2, "<comparison predicate>", Subclause 7.3, "<overlaps predicate>", Subclause 7.4, "<precedes predicate>", Subclause 7.5, "<contains predicate>", or Subclause 7.6, "<meets predicate>", as appropriate.

7.2 <comparison predicate>

Function

Specify a comparison of two row values.

Format

No additional Format items.

Syntax Rules

1) If the data type of the <row value constructor>s is a period data type, then <comp op> shall be <equals operator>.

Access Rules

No additional Access Rules.

General Rules

1) Case:

   a) If the data type of <row value expression 1> is PERIOD, then:

      i) Let *X* and *Y* be any two corresponding <row value constructor element>s. Let *XV* and *YV* be the values represented by *X* and *Y*, respectively.

      ii) Let *XS* and *XE* be the beginning and ending delimiting timestamps, respectively, of *XV*.

      iii) Let *YS* and *YE* be the beginning and ending delimiting timestamps, respectively, of *YV*.

      iv) "*X* = *Y*" is true if and only if *XS* and *YS* are equal and *XE* and *YE* are equal.

7.3 <overlaps predicate>

****Editor's Note****
This specification of the <overlaps predicate> appears to be incompatible with SQL-92's <overlaps predicate>. If this is a correct interpretation, then this specification must be revisited to make it compatible.

Function

Specify a test for an overlap between two periods.

Format

No additional Format items.

Syntax Rules

1) Let *T1* be the type of <row value expression 1> and let *T2* be the type of <row value expression 2>.

2) *T1* and *T2* shall be either PERIOD or TIMESTAMP. One of T1 and T2 shall be of data type PERIOD.

3) The precision of *T1* and *T2* shall be identical.

Access Rules

No additional Access Rules.

General Rules

1) Case:

   a) If the data type of <row value expression 1> is TIMESTAMP, then let *B1* and *E1* both be the value of <row value expression 1>.

   b) Otherwise, let *P1* be the value of <row value expression 1> and let *B1* and *E1* be the beginning and ending delimiting timestamps, respectively, of *P1*.

2) Case:

   a) If the data type of <row value expression 2> is TIMESTAMP, then let *B2* and *E2* both be the value of <row value expression 2>.

   b) Otherwise, let *P2* be the value of <row value expression 1> and let *B2* and *E2* be the beginning and ending delimiting timestampes, respectively, of *P2*.

3) The value of the <overlaps predicate> is the value of

   B1 <= *E2* AND *B2* <= *E1*

7.4 <precedes predicate>

Function

Specify a test for one period or timestamp preceding a second period or timestamp.

Format

```
<precedes predicate> ::=
       <row value expression 1> PRECEDES <row value expression 2>
```

Syntax Rules

1) Let *T1* be the type of <row value expression 1> and let *T2* be the type of <row value expression 2>.

2) *T1* and *T2* shall be either a period or TIMESTAMP.

3) The precision of *T1* and *T2* shall be identical.

Access Rules

None.

General Rules

1) Case:

   a) If the data type of <row value expression 1> is TIMESTAMP, then let *E1* be the value of <row value expression 1>.

   b) Otherwise, let *P1* be the value of <row value expression 1> and let *E1* be the ending delimiting timestamp of *P1*.

2) Case:

   a) If the data type of <row value expression 2> is TIMESTAMP, then let *B2* be the value of <row value expression 2>.

   b) Otherwise, let *P2* be the value of <row value expression 2> and let *B2* be the beginning delimiting timestamp of *P2*.

3) The value of the <precedes predicate> is the value of

   E1 < B2

7.5 <contains predicate>

Function

Specify a test for a period containing another period or a TIMESTAMP.

Format

```
<contains predicate> ::=
      <row value expression 1> CONTAINS <row value expression 2>
```

Syntax Rules

1) Let *T1* be the type of <row value expression 1> and let *T2* be the type of <row value expression 2>.

2) *T1* and *T2* shall be either a period or TIMESTAMP.

3) The precision of *T1* and *T2* shall be identical.

Access Rules

None.

General Rules

1) Let *P1* be the value of <row value expression 1> and let *B1* and *E1* be the beginning and ending timestamps, respectively, of *P1*.

2) Case:

   a) If the data type of <row value expression 2> is TIMESTAMP, then let *B2* and *E2* both be the value of <row value expression 2>.

   b) Otherwise, let *P2* be the value of <row value expression 1> and let *B2* and *E2* be the beginning and ending delimiting timestamps, respectively, of *P2*.

3) The value of the <contains predicate> is the value of

   B1 <= *B2* AND *E2* <= *E1*

7.6 <meets predicate>

Function

Specify a test for one period or timestamp adjoining another period or timestamp.

Format

```
<meets predicate> ::=
       <row value expression 1> MEETS <row value expression 2>
```

Syntax Rules

1) Let *T1* be the type of <row value expression 1> and let *T2* be the type of <row value expression 2>.

2) *T1* and *T2* shall be either a period or TIMESTAMP.

3) The precision of *T1* and *T2* shall be identical.

Access Rules

None.

General Rules

1) Case:

   a) If the data type of <row value expression 1> is TIMESTAMP, then let *E1* be the value of <row value expression 1>.

   b) Otherwise, let *P1* be the value of <row value expression 1> and let *E1* be the ending delimiting timestamp of *P1*.

2) Case:

   a) If the data type of <row value expression 2> is TIMESTAMP, then let *B2* be the value of <row value expression 2>.

   b) Otherwise, let *P2* be the value of <row value expression 2> and let *B2* be the beginning delimiting timestamp of *P2*.

3) Let *P* be the precision of <row value expression 1>.

4) For $P \geq$, the value of the <meets predicate> is the value of

   Case:

   a) If $P > 1$, then:

      (*E1* + INTERVAL '00:00:00.0...01' HOUR TO SECOND(*P*) = *E2*)

      where there are $P - 1$ zeros to the right of the decimal point.

b) Otherwise:

(*E1* + INTERVAL '00:00:01' HOUR TO SECOND = *E2*)

8 Data assignment rules

8.1 Retrieval assignment

Function

Specify rules for value assignments that retrieve SQL-data.

Syntax Rules

1) Let *T* and *V* be a *TARGET* and *VALUE* specified in an application of this Subclause.

2) 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) 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.

8.2 Store assignment

Function

Specify rules for value assignments that store SQL-data.

Syntax Rules

1) Let *T* and *V* be a *TARGET* and *VALUE* specified in an application of this Subclause.

2) 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) 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.

8.3 Set operation result data types

Function

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

Syntax Rules

1) Let *DTS* be a set of data types specified in an application of this Subclause.

2) If any data type in *DTS* is a period data type, then each data type in *DTS* shall be the same period data type. The result data type is the same period data type.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

9 Schema definition and manipulation

9.1 <default clause>

Function

Specify the default for a column or domain.

Format

No additional Format items.

Syntax Rules

1) 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) 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>.

10 Information Schema and Definition Schema

10.1 Information Schema

There are no changes or additions to this Part of this International Standard.

10.2 Definition Schema

10.2.1 DATA_TYPE_DESCRIPTOR base table

Function

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 represention of the data type descriptors.

Definition

```
CREATE TABLE DATA_TYPE_DESCRIPTOR
       (
       TABLE_OR_DOMAIN_CATALOG            INFORMATION_SCHEMA.SQL_IDENTIFIER,
       TABLE_OR_DOMAIN_SCHEMA             INFORMATION_SCHEMA.SQL_IDENTIFIER,
       TABLE_OR_DOMAIN_NAME               INFORMATION_SCHEMA.SQL_IDENTIFIER,
       COLUMN_NAME                        INFORMATION_SCHEMA.SQL_IDENTIFIER,
       DATA_TYPE                          INFORMATION_SCHEMA.CHARACTER_DATA
         CONSTRAINT TABLE_OR_DOMAIN_DATA_TYPE_NOT_NULL NOT NULL,
       CHARACTER_MAXIMUM_LENGTH           INFORMATION_SCHEMA.CARDINAL_NUMBER,
       CHARACTER_OCTET_LENGTH             INFORMATION_SCHEMA.CARDINAL_NUMBER,
       COLLATION_CATALOG                  INFORMATION_SCHEMA.SQL_IDENTIFIER,
       COLLATION_SCHEMA                   INFORMATION_SCHEMA.SQL_IDENTIFIER,
       COLLATION_NAME                     INFORMATION_SCHEMA.SQL_IDENTIFIER,
       NUMERIC_PRECISION                  INFORMATION_SCHEMA.CARDINAL_NUMBER,
       NUMERIC_PRECISION_RADIX            INFORMATION_SCHEMA.CARDINAL_NUMBER,
       NUMERIC_SCALE                      INFORMATION_SCHEMA.CARDINAL_NUMBER,
       DATETIME_PRECISION                 INFORMATION_SCHEMA.CARDINAL_NUMBER,
       INTERVAL_CODE                      INFORMATION_SCHEMA.CHARACTER_DATA,
       INTERVAL_PRECISION                 INFORMATION_SCHEMA.CARDINAL_NUMBER,
       ABSTRACT_DATA_TYPE_CATALOG         INFORMATION_SCHEMA.SQL_IDENTIFIER,
       ABSTRACT_DATA_TYPE_SCHEMA          INFORMATION_SCHEMA.SQL_IDENTIFIER,
       ABSTRACT_DATA_TYPE_NAME            INFORMATION_SCHEMA.SQL_IDENTIFIER,

       CONSTRAINT TABLE_OR_DOMAIN_CHECK_COMBINATIONS
         CHECK ( DATA_TYPE IN ( 'CHARACTER', 'CHARACTER VARYING',
                                'BIT', 'BIT VARYING' )
               AND ( CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
                     COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME )
                       IS NOT NULL
               AND ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX,
                     NUMERIC_SCALE, DATETIME_PRECISION,
                     ABSTRACT_DATA_TYPE_CATALOG, ABSTRACT_DATA_TYPE_SCHEMA,
                     ABSTRACT_DATA_TYPE_NAME ) IS NULL
               AND ( INTERVAL_CODE, INTERVAL_PRECISION )
                     IS NULL
         OR
```

```
        DATA_TYPE IN ( 'INTEGER', 'SMALLINT' )
        AND ( CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
              COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME )
                  IS NULL
        AND NUMERIC_PRECISION_RADIX IN ( 2, 10 ) )
        AND NUMERIC_PRECISION IS NOT NULL
        AND NUMERIC_SCALE = 0
        AND DATETIME_PRECISION IS NULL
        AND ( INTERVAL_CODE, INTERVAL_PRECISION ) IS NULL
OR
        DATA_TYPE IN ( 'NUMERIC', 'DECIMAL' )
        AND ( CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
              COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME )
                       IS NULL
        AND NUMERIC_PRECISION_RADIX = 10
        AND ( NUMERIC_PRECISION, NUMERIC_SCALE ) IS NOT NULL
        AND DATETIME_PRECISION IS NULL
        AND ( INTERVAL_CODE, INTERVAL_PRECISION ) IS NULL
        AND ( INTERVAL_CODE, INTERVAL_PRECISION )
              IS NULL
OR
        DATA_TYPE IN ( 'REAL', 'DOUBLE PRECISION', 'FLOAT' )
        AND ( CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
              COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME )
                IS NULL
        AND NUMERIC_PRECISION IS NOT NULL
        AND NUMERIC_PRECISION_RADIX = 2
        AND NUMERIC_SCALE IS NULL
        AND DATETIME_PRECISION IS NULL
        AND ( ABSTRACT_DATA_TYPE_CATALOG, ABSTRACT_DATA_TYPE_SCHEMA,
                                ABSTRACT_DATA_TYPE_NAME ) IS NULL
        AND ( INTERVAL_CODE, INTERVAL_PRECISION )
              IS NULL
OR
        DATA_TYPE IN ( 'DATE', 'TIME', 'TIMESTAMP',
                       'TIME WITH TIME ZONE', 'TIMESTAMP WITH TIME ZONE',
                       'PERIOD', "PERIOD WITH TIME ZONE' )
        AND ( CHARACTER_MAXIMUM_LENGTH,
              CHARACTER_OCTET_LENGTH,
              COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME )
                IS NULL
        AND ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX ) IS NOT NULL
        AND NUMERIC_SCALE IS NULL
        AND DATETIME_PRECISION IS NOT NULL
        AND ( ABSTRACT_DATA_TYPE_CATALOG, ABSTRACT_DATA_TYPE_SCHEMA,
              ABSTRACT_DATA_TYPE_NAME ) IS NULL
        AND ( INTERVAL_CODE, INTERVAL_PRECISION )
              IS NULL
OR
        DATA_TYPE = 'INTERVAL'
        AND ( CHARACTER_MAXIMUM_LENGTH,
              CHARACTER_OCTET_LENGTH,
              COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME )
                IS NULL
        AND ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX ) IS NOT NULL
        AND NUMERIC_SCALE IS NULL
        AND DATETIME_PRECISION IS NOT NULL
        AND ( ABSTRACT_DATA_TYPE_CATALOG, ABSTRACT_DATA_TYPE_SCHEMA,
              ABSTRACT_DATA_TYPE_NAME ) IS NULL
        AND INTERVAL_CODE IN
              ( 'YEAR', 'MONTH', 'DAY', 'HOUR',
                'MINUTE', 'SECOND', 'YEAR TO MONTH',
                'DAY TO HOUR', 'DAY TO MINUTE',
                'DAY TO SECOND', 'HOUR TO MINUTE',
```

```
              'HOUR TO SECOND', 'MINUTE TO SECOND' )
        AND INTERVAL_PRECISION
          IS NOT NULL
  OR
        DATA_TYPE = 'BOOLEAN'
        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 ( ABSTRACT_DATA_TYPE_CATALOG, ABSTRACT_DATA_TYPE_SCHEMA,
              ABSTRACT_DATA_TYPE_NAME ) IS NULL
        AND ( INTERVAL_CODE, INTERVAL_PRECISION )
              IS NULL
  OR
        DATA_TYPE = 'ENUMERATED'
        AND ( CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
              COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME,
              NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX,
              NUMERIC_SCALE, DATETIME_PRECISION,
              ABSTRACT_DATA_TYPE_CATALOG, ABSTRACT_DATA_TYPE_SCHEMA,
              ABSTRACT_DATA_TYPE_NAME, INTERVAL_TYPE, INTERVAL_PRECISION )
                IS NULL
  OR
        DATA_TYPE = 'USER_DEFINED'
        AND ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX,
              NUMERIC_SCALE, DATETIME_PRECISION, CHARACTER_OCTET_LENGTH,
              CHARACTER_MAXIMUM_LENGTH, INTERVAL_CODE, INTERVAL_PRECISION )
                IS NULL
  OR
        DATA_TYPE NOT IN
            ( 'CHARACTER', 'CHARACTER VARYING',
              'BIT', 'BIT VARYING',
              'INTEGER', 'SMALLINT', 'NUMERIC', 'DECIMAL',
              'REAL', 'DOUBLE PRECISION', 'FLOAT',
              'DATE', 'TIME', 'TIMESTAMP', 'PERIOD',
              'INTERVAL', 'BOOLEAN', 'ENUMERATED', 'USER_DEFINED' )
  ),

CONSTRAINT DATA_TYPE_DESCRIPTOR_PRIMARY_KEY
  PRIMARY KEY ( TABLE_OR_DOMAIN_CATALOG, TABLE_OR_DOMAIN_SCHEMA,
                  TABLE_OR_DOMAIN_NAME, COLUMN_NAME ),

CONSTRAINT DATA_TYPE_CHECK_REFERENCES_COLLATION
    CHECK ( COLLATION_CATALOG
              <> ANY ( SELECT CATALOG_NAME FROM SCHEMATA )
        OR
           ( COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME ) IN
           ( SELECT COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME
             FROM COLLATIONS ) ),

CONSTRAINT DATA_TYPE_DESCRIPTOR_FOREIGN_KEY_SCHEMATA
  FOREIGN KEY ( ABSTRACT_DATA_TYPE_CATALOG, ABSTRACT_DATA_TYPE_SCHEMA)
  REFERENCES SCHEMATA,

CONSTRAINT DATA_TYPE_DESCRIPTOR_CHECK_USED
  CHECK ( (TABLE_OR_DOMAIN_CATALOG, TABLE_OR_DOMAIN_SCHEMA,
             TABLE_OR_DOMAIN_NAME, COLUMN_NAME)
    IN (
        SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME
            FROM COLUMNS
        UNION
```

```
            SELECT DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME, ''
                FROM DOMAINS )
    )
```

Description

No additional Description items.

11 Status codes

11.1 SQLSTATE

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

Table 2—SQLSTATE class and subclass values

Category	Condition	Class	Subcondition	Subclass
X	data exception	22	(*no subclass*)	000
			period field overflow	030
			invalid period value literal	031
			invalid period format	032

12 Conformance

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

1) A conformance claim to
   SQL2 ISO/IEC 9075:1992, as specified in Subclause 23.2, "Claims of conformance".
   SQL3 Part 2 of this International Standard, in Clause 20, "Conformance".
   SQL4 Part 2 of this International Standard, in Clause 20, "Conformance".

2) The definitions for all elements and actions that this Part of this International Standard specifies as implementation-defined.

3) Other items as required.

Annex A
(informative)

Implementation-defined elements

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

The term *implementatation-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.

Annex B
(informative)

Implementation-dependent elements

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

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

There are no additional implementation-dependent elements.

Annex C
(informative)

Deprecated features

It is intendend that the following features will be removed at a later date frm a revised version of this Part of this International Standard:

There are no additional deprecated items.

Annex D
(informative)

Incompatibilities with ISO/IEC 9075:1992

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

   — MEETS

   — PERIOD

   — PRECEDES

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.