June 30, 2022

ISO/IEC JTC1/SC32/WG3

ANSI INCITS DM32

Database

Title: Introduction of multi-character string trimming functions

Author: Gerald Venzl

Project: SQL: 202X-CD

Status: Proposal

References:

[Foundation CD] WG3:W05-003R1 (DM32-2020-00412R1) *(ISO Committee Draft) Database Language SQL - Part 2: Foundation (SQL/Foundation),* Jim Melton (editor), November 2020.

[Foundation IWD] WG3:BER-003 (DM32-2022-00223) *(ISO Committee Draft) Database Language SQL - Part 2: Foundation (SQL/Foundation),* Jim Melton (editor), May 2022.

[CD comments] WG3:BER-009 (DM32-2022-00243) SC32/WG3 CD 9075 Consolidated Comments, Jim Melton (editor)

1 Introduction

This paper addresses the comment:

> #49 - P02-USA-120
> 1-Major Technical
> P02-06.32, <string value function>
>
> In addition to the TRIM function, existing SQL-implementations provide RTRIM and LTRIM functions. Rather than being just syntactic sugar/shorthand for trimming a single character on the right or left side, respectively, of the string value, these SQL-implementations allow the user to specify more than one character to be trimmed from the value.
>
> For example,
>
> RTRIM('<=====>Peter<=====>', '<>=')
>
> Returns:
>
> <=====>Peter
>
> The SQL standard should provide this functionality as well.
>
> Solution: None provided with comment.

1.1 Rationale

While the already existing TRIM function allows users to trim a given leading and/or trailing character one or multiple times in a string, it does not allow users to specify a multitude of characters to be removed. Hence to remove multiple characters potentially leading or trailing a string, users currently have to invoke the existing TRIM function numerous times for every character. To avoid having to go through nested function calls for each character, a new trimming function should be introduced that can handle this scenario more elegantly.

This paper proposes three new string trimming functions for the SQL standard:

- LTRIM to trim leading characters of the string.
- RTRIM to trim trailing characters of the string.
- BTRIM to trim the leading and trailing characters of the string.

1.2 Examples

With the new trim functions, a user can remove a set of leading and/or trailing characters in one go, as illustrated below.

1.2.1 Remove a character sequence from a string

In this example, the user wishes to remove the character sequence “<=====>” from the string:

```
SELECT RTRIM('Gerald Venzl<=====>', '<=====>') AS result FROM dual;

RESULT
------------
Gerald Venzl
```

```
SELECT LTRIM('<=====>Gerald Venzl', '<=====>') AS result FROM dual;

RESULT
------------
Gerald Venzl
```

1.2.2 Remove any leading/trailing characters from a string

Because these trim functions remove leading/trailing occurrences of any of the trim characters, the sequence of the trim characters does not have to match exactly with the leading/trailing characters in the trim source. The same example above can be rewritten as follows, note the shorter trim characters sequence:

```
SELECT RTRIM('Gerald Venzl<=====>', '<=>') AS result FROM dual;

RESULT
------------
Gerald Venzl



SELECT LTRIM('<=====>Gerald Venzl', '<=>') AS result FROM dual;

RESULT
------------
Gerald Venzl
```

Because these trim functions remove any leading/trailing occurrence of any of the trim characters, the trim characters do not even have to have the same sequence as in the trim source. This is particularly useful to remove multiple leading/trailing characters without having to inspect the trim source first:

```
SELECT RTRIM('Gerald Venzl<=====>', '><=') AS result FROM dual;

RESULT
------------
Gerald Venzl



SELECT LTRIM('<=====>Gerald Venzl', '><=') AS result FROM dual;

RESULT
------------
Gerald Venzl
```

The BTRIM function combines the execution of LTRIM and RTRIM and removes leading *and* trailing characters in the same invocation. Consider the following example (executed on Postgres) where a user wishes to remove leading and trailing characters from a string. This can be accomplished via a chained invocation of both LTRIM and RTRIM (or vice versa):

```
test=> SELECT RTRIM(LTRIM('<=====>Gerald Venzl<=====>', '><='), '><=') AS result;
    result
--------------
 Gerald Venzl
(1 row)
```

Using the BTRIM function, such nesting of LTRIM and RTRIM is not necessary:

```
test=> SELECT BTRIM('<=====>Gerald Venzl<=====>', '><=') AS result;
    result
--------------
 Gerald Venzl
(1 row)
```

1.3 Conventions

SMALLCAPS	denote numbered editorial instructions;
New Text	New text is identified by using a red bold font;
~~Deleted Text~~	Deleted text is identified by using a blue strikethrough font;
Plain	denotes existing text to be retained;
[Note]	denotes notes to the editor.

2 Proposal

2.1 Proposed changes for [Foundation IWD]

2.1.1 Changes to 4.3.3.2 Operators that operate on character strings and return character strings

1. CHANGE THE FOLLOWING PARAGRAPH AS SHOWN HERE:

   <**single-character** trim function> is a function that returns its first string argument with leading and/or trailing pad characters removed. The second argument indicates whether leading, or trailing, or both leading and trailing pad characters shall be removed. The third argument specifies the pad character that is to be removed.

2. ADD NEW PARAGRAPH AS SHOWN HERE:

   <multi-character trim function> is a function that returns its first string argument with leading and/or trailing pad characters removed. The second argument specifies the pad characters that are to be removed. In contrast to the <single-character trim function>, this function accepts a sequence of characters as the second argument and will remove all leading and/or trailing occurrences of any of these characters, regardless of their ordering.

2.1.2 Changes to 4.4.3.1 Operators that operate on binary strings and return binary strings

1. CHANGE THE FOLLOWING PARAGRAPH AS SHOWN HERE:

   <binary trim function> is a function identical in syntax and semantics to <**single-character** trim function> except that its arguments and the returned value are all binary strings.

2.1.3 Changes to 5.2 <token>

2. EXPAND <RESERVED WORD> ::= WITH ADDITIONAL WORDS AS SHOWN HERE

```
| BEGIN | BEGIN_FRAME | BEGIN_PARTITION | BETWEEN | BIGINT | BINARY
| BLOB | BOOLEAN | BOTH | BTRIM | BY

| LAG | LANGUAGE | LARGE | LAST_VALUE | LATERAL | LEAD | LEADING | LEAST | LEFT
| LIKE | LIKE_REGEX | LISTAGG | LN | LOCAL | LOCALTIME | LOCALTIMESTAMP | LOG
| LOG10 | LOWER | LPAD | LTRIM

| RANGE | RANK | READS | REAL | RECURSIVE | REF | REFERENCES | REFERENCING
| REGR_AVGX | REGR_AVGY | REGR_COUNT | REGR_INTERCEPT | REGR_R2 | REGR_SLOPE
| REGR_SXX | REGR_SXY | REGR_SYY | RELEASE | RESULT | RETURN | RETURNS
| REVOKE | RIGHT | ROLLBACK | ROLLUP | ROW | ROW_NUMBER | ROWS | RPAD | RTRIM |
RUNNING
```

2.1.4 Changes to 6.33 <string value function>

1. MODIFY THE FORMAT AS SHOWN HERE:

<trim function> ::=
<single-character trim function>
| <multi-character trim function>

<single-character trim function> ::=
TRIM <left paren> <trim operands> <right paren>

<multi-character trim function> ::=
{ BTRIM | LTRIM | RTRIM } <left paren> <trim source> [<comma> <trim character>] <right paren>

2. ADD NEW SR IMMEDIATELY BEFORE SR 13 AS SHOWN HERE:

13.0) The declared type of <trim function> is the declared type of the immediately contained <single-character trim function>, or <multi-character trim function>.

3. CHANGE SR 13 AS SHOWN HERE:

13) If <**single-character** trim function> is specified, then

a) Case:

i) If FROM is specified, then:

1) Either <trim specification> or <trim character> or both shall be specified.

2) If <trim specification> is not specified, then BOTH is implicit.

3) If <trim character> is not specified, then <space> is implicit.

ii) Otherwise, let *SRC* be <trim source>. `TRIM ( SRC )` is equivalent to `TRIM ( BOTH ' ' FROM SRC )`.

b) Case:

i) If the declared type of <character value expression> **simply contained in <trim source>** is fixed-length character string or variable-length character string, then the declared type of the <**single-character** trim function> is variable- length character string with maximum length equal to the length or maximum length of the <trim source>.

Note to the reader: tightening of rule and aligning with <multi-character trim function>.

ii) Otherwise, the declared type of the <**single-character** trim function> is a character large object type with maximum length equal to the maximum length of the <trim source>.

c) If a <trim character> is specified, then <trim character> and <trim source> shall be comparable.

d) The character set and collation of the <**single-character** trim function> are those of the <trim source>.

4. INSERT NEW SYNTAX RULE IMMEDIATELY AFTER SYNTAX RULE 13 AS SHOWN HERE:

13.1) If <multi-character trim function> *MTF* is specified, then:

a) Case:

i) If the declared type of <character value expression> simply contained in <trim source> *TS* is fixed-length character string or variable-length character string, then the declared type of *MTF* is variable- length character string with maximum length equal to the length or maximum length of *TS*.

ii) Otherwise, the declared type of *MTF* is a character large object type with maximum length equal to the maximum length of *TS*.

b) Case:

i) If <trim character> *TC* is specified, then:

A. *TC* and *TS* shall be comparable.

B. If BTRIM is specified, then:

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

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

c. `BTRIM ( TS, TC )` is equivalent to `RTRIM ( LTRIM ( TS, TC ), TC )`

ii) Otherwise:

A. Let *TC* be <space>.

B. If BTRIM is specified, then `BTRIM ( TS )` is equivalent to `RTRIM ( LTRIM ( TS ) )`

c) The character set and collation of *MTF* are those of *TS*.

5. INSERT NEW GENERAL RULE IMMEDIATE BEFORE GR 11

11.0) The result of <trim function> is the result of the immediately contained <single-character trim function> or <multi-character trim function>.

6. MODIFY GENERAL RULE 11 AS SHOWN HERE:

11) If <**single-character** trim function> is specified, then:

a) Let *S* be the value of the <trim source>.

b) ~~If <trim character> is specified, then let~~ **Let** *SC* be the value of <trim character>.~~; otherwise, let *SC* be <space>.~~

Note to the reader: since syntax rule 13) a) always provides a default for <trim character> if not specified, the GRs can rely on the <trim character> being specified.

c) If at least one of *S* and *SC* is the null value, then the result of the <**single-character** trim function> is the null value and **no further General Rules of this Subclause are applied**.

Note to the reader: tightening of rules.

d) If the length in characters of *SC* is not 1 (one), then an exception condition is raised: *data exception — trim error (22027)*.

e) Case:

i) If BOTH is specified or if no <trim specification> is specified, then the result of the <**single-character** trim function> is the value of *S* with any leading or trailing characters equal to *SC* removed.

ii) If TRAILING is specified, then the result of the <**single-character** trim function> is the value of *S* with any trailing characters equal to *SC* removed.

iii) If LEADING is specified, then the result of the <**single-character** trim function> is the value of *S* with any leading characters equal to *SC* removed.

7. ADD NEW GENERAL RULE IMMEDIATELY AFTER GR 11 AS SHOWN HERE:

11.1) If <multi-character trim function> *MTF* is specified, then:
- **a) Let *S* be the value of *TS*.**
- **b) Let *SC* be the value of *TC*.**
- **c) If at least one of *S* and *SC* is the null value, then the result of *MTF* is the null value and no further General Rules of this Subclause are applied.**
- **d) If *S* is the zero-length character string or *SC* is the zero-length character string, then the result of *MTF* is *S* and no further General Rules of this Subclause are applied.**
- **e) Case:**
  - **i) If RTRIM is specified:**
    - **A. Let *P* be the length of *S*.**
    - **B. Let *C* be the *P*-th character of *S*.**
    - **C. While *C* is contained in *SC*, *do:***
      - **A. Let *P* be *P* – 1**
      - **B. Let *C* be the *P*-th character of *S*.**
    - **D. The result of *MTF* is SUBSTRING (*TS* FROM 1 FOR *P*)**
  - **ii) If LTRIM is specified:**
    - **A. Let *P* be 1 (one).**
    - **B. Let *C* be the *P*-th character of *S*.**
    - **C. While *C* is contained in *SC*, *do:***
      - **A. Let *P* be *P* + 1**
      - **B. Let *C* be the *P*-th character of *S*.**
    - **D. The result of *MTF* is SUBSTRING (*TS* FROM *P*)**

8. ADD NEW CONFORMANCE RULE AS SHOWN HERE:

N) Without Feature FNNN, "Multi-character TRIM functions", conforming SQL language shall not contain a <multi-character trim function>.

2.1.5 Changes to Annex D, Table 45 – Feature taxonomy for optional features

1. ADD NEW ROW:

N | FNNN | Multi-character TRIM functions

2.1.6 Changes to Annex F, 1)

1. ADD BTRIM, LTRIM AND RTRIM TO <RESERVED WORDS> AS SHOWN BELOW:

— ANY_VALUE

— BTRIM

— GREATEST

— JSON_SCALAR

— JSON_SERIALIZE

— LEAST

— LPAD

— LTRIM

— RPAD

— RTRIM

2.1.7 Changes to Annex H, Table 46 – Feature taxonomy and definition for mandatory features

1. CHANGE ROW 17 AS SHOWN HERE:

17	E021-09	TRIM function	— Subclause 6.33, “<string value function>”: The <**single-character** trim function>

2.2 Proposed changes for [Foundation CD]

2.2.1 Apply the same changes as in section 2.1

3 Checklist

Interactions with other concurrent proposals identified and editorial assistance given	N/A
Concepts	Yes
Access Rules	N/A
Conformance Rules	Yes
Lists of SQL- statements by category	N/A
Table of identifiers used by diagnostics statements	N/A
Collation coercibility for character strings	N/A
Closing Possible Problems	Yes
Any new Possible Problems clearly identified	N/A
Reserved and non- reserved keywords	Yes
SQLSTATE tables and Ada package	N/A
Information and Definition Schemas	N/A
Implementation- defined and –dependent Annexes	N/A
Incompatibilities Annex	Yes
Embedded SQL and host language implications	N/A
Dynamic SQL issues: including descriptor areas	N/A
PSM impact	None
Schemata impact	None
XML impact	None
PGQ impact	None

- End of paper -