Correct <cast error behavior>

Author: Vik Fearing (SIS)
Source: Individual Expert Contribution
Status: Change Proposal
Date: April 2, 2026
Document: ISO/IEC JTC 1/SC 32/WG 3:BMA-014 draft-01

1. Abstract

The current grammar for <cast error behavior> provides a standalone NULL alternative alongside DEFAULT <value expression>. Because NULL is an <implicitly typed value specification> and not a <value expression>, the form DEFAULT NULL ON CONVERSION ERROR is a syntax error. The same problem applies to empty collection literals such as ARRAY[] and MULTISET[].

This paper removes the standalone NULL alternative and replaces DEFAULT <value expression> with DEFAULT <cast error default>, where the new production accepts either a <value expression> or an <implicitly typed value specification>. The result is a uniform syntax: DEFAULT *value* ON CONVERSION ERROR, covering nulls, empty collections, literals, and general expressions.

2. References

[Foundation] ISO/IEC 9075-2, SQL/Foundation (IWD)

[CMN-027r3] Vik Fearing, “Extending CAST” (change proposal, accepted)

3. Discussion

3.1. Background

The error handling clause for CAST was introduced by [CMN-027r3], which added three alternatives to <cast error behavior>: ERROR, NULL, and DEFAULT <value expression>. The NULL alternative was included as syntactic sugar, described in the paper as equivalent to DEFAULT CAST(NULL AS *type*) ON CONVERSION ERROR.

3.2. The Problem

Subclause 6.13, “<cast specification>”, defines the error handling clause for CAST:

```
<cast error behavior> ::=
    ERROR
  | NULL
  | DEFAULT <value expression>
```

The `NULL` alternative is a special case: it is the only implicitly typed value specification that has its own keyword in this production. The more general `DEFAULT` branch requires a `<value expression>`, which excludes all implicitly typed value specifications.

As a result, these are syntax errors:

```
-- desired: default to an empty array on error
CAST(x AS INTEGER ARRAY[10] DEFAULT ARRAY[] ON CONVERSION ERROR)

-- desired: default to null on error (using DEFAULT form)
CAST(x AS INTEGER DEFAULT NULL ON CONVERSION ERROR)
```

The first has no workaround short of providing a non-empty array literal or a subquery. The second can be written as `NULL ON CONVERSION ERROR`, but only because the grammar has a dedicated alternative for it.

3.3. The Fix

Replace the ad-hoc `NULL` keyword with a uniform `DEFAULT` branch that accepts both value expressions and implicitly typed value specifications. A new production `<cast error default>` captures this union:

```
<cast error behavior> ::=
    ERROR
  | DEFAULT <cast error default>

<cast error default> ::=
    <value expression>
  | <implicitly typed value specification>
```

Under this grammar:

- `DEFAULT NULL ON CONVERSION ERROR` works (via `<implicitly typed value specification>`).
- `DEFAULT ARRAY[] ON CONVERSION ERROR` works (via `<implicitly typed value specification>`).
- `DEFAULT 0 ON CONVERSION ERROR` continues to work (literal, via `<value expression>`).
- `DEFAULT some_expr ON CONVERSION ERROR` continues to work (general expression).

4. Change Proposal Conventions

The following conventions are used in this change proposal:

plain text	current text to be retained
bold red	new text to be inserted
~~strikeout blue~~	current text to be deleted
[bold bracketed orange]	instruction to the editors
[italic bracketed green]	information for the proposal reader
italic purple	first introduction of a symbol
italic	subsequent use of a symbol

5. Change Proposal, [Foundation]

[The following changes are to be applied to [Foundation]]

5.1. Subclause 6.13, "<cast specification>"

[Modify the Format]

```
<cast error behavior> ::=
    ERROR
  | NULL
  | DEFAULT <value expression>
  | DEFAULT <cast error default>

<cast error default> ::=
    <value expression>
  | <implicitly typed value specification>
```

[Modify Syntax Rule 6)]

6) If *CEB* specifies a ~~<value expression>~~ **<cast error default>** *CED* ~~*DVE*~~, then ~~the declared type of *DVE* shall be assignable to *TD*.~~

a) If *CED* is an <implicitly typed value specification>, then:

i) If *CED* is a <null specification>, then its declared type is *TD*.

ii) Otherwise, *TD* shall be a collection type. The declared type of *CED* is determined by applying Subclause 6.5, "<contextually typed value specification>", Syntax Rule 1) with element type taken from *TD*.

b) Otherwise, let *DVE* be *CED*. The declared type of *DVE* shall be assignable to *TD*.

[Modify General Rule 24)b)]

24) Case:

a) If *TEMPEC* is *successful completion (00000)*, then the result of *CS* is *TV*.

b) Otherwise,

Case:

i) If *CEB* is ERROR, then the exception condition *TEMPEC* is raised and no further General Rules of this subclause are applied.

~~ii) If *CEB* is NULL, then the result of *CS* is the null value.~~

~~iii)~~**ii)** If *CEB* is DEFAULT, ~~then let *DVE* be the <value expression> immediately contained in *CEB*. The result of *CS* is~~ **then let *CED* be the <cast error default> immediately contained in *CEB*.**

Case:

1) If *CED* is an <implicitly typed value specification>, then:

A) If *CED* is a <null specification>, then the result of *CS* is the null value.

B) Otherwise, the result of *CS* is an empty collection of declared type *TD*.

2) Otherwise, let *DVE* be *CED*. The result of *CS* is

```
CAST ( DVE AS TD ERROR ON CONVERSION ERROR )
```

[Modify Conformance Rule 7)]

7) Without Feature F205, "General value expression in ON CONVERSION ERROR clause", ~~the <value expression> contained in <cast error behavior>~~ **if the <cast error default> contained in <cast error behavior> is a <value expression>, then it** shall be a <literal> that can be cast to the <cast target> without raising an exception condition according to the General Rules of Subclause 6.13, "<cast specification>".

6. Incompatibilities

The standalone NULL keyword is removed from <cast error behavior>. Conforming SQL language that contains NULL ON CONVERSION ERROR must be rewritten to DEFAULT NULL ON CONVERSION ERROR.

Because [CMN-027r3] was accepted after the SQL:2023 release, <cast error behavior> has never appeared in a published edition of the standard. No Annex F entry is needed.

7. Checklist

Item	Status
Interactions with other concurrent proposals identified and editorial assistance given.	None known
Concepts.	Yes
Access Rules.	N/A
Conformance Rules, including the relevant Annexes.	Yes
Check that any removed non-terminals, subclauses, defined terms, and the like are no longer referenced.	N/A
BNF production split affecting immediate containment.	Yes
Lists of SQL-statements by category.	N/A
Collation coercibility determination for changes related to character strings.	N/A
Closing Possible Problems or Language Opportunity when a proposal resolves them.	N/A
Any new Possible Problems clearly identified.	N/A
Reserved and non-reserved keywords.	N/A
Implementation-defined and -dependent features clearly identified, and descriptions provided.	N/A
Information and Definition Schemas.	N/A
Incompatibilities Annex (for 2nd and subsequent editions).	N/A
Table of identifiers used by diagnostics statements.	N/A
Embedded SQL bindings and host language implications.	N/A
Dynamic SQL issues, including Dynamic descriptor areas.	N/A
CLI impact.	N/A
PSM impact.	N/A
MED impact.	N/A
OLB impact.	N/A
Schemata impact.	N/A
JRT impact.	N/A

Item	Status
XML impact	N/A
MDA impact.	N/A
PGQ impact.	N/A
Impact on GQL for PGQ changes.	N/A
Any Guidance IS impact	N/A

end of paper