sql_variant (Transact-SQL)

09/12/2017 • 3 minutes to read •

In this article

APPLIES TO: ✓ SQL Server ✓ Azure SQL Database ✗ Azure Synapse Analytics (SQL DW) ✗ Parallel Data Warehouse

A data type that stores values of various SQL Server-supported data types.

Transact-SQL Syntax Conventions

Syntax

SQL Copy

```
sql_variant
```

Remarks

sql_variant can be used in columns, parameters, variables, and the return values of user-defined functions. **sql_variant** enables these database objects to support values of other data types.

A column of type **sql_variant** may contain rows of different data types. For example, a column defined as **sql_variant** can store **int**, **binary**, and **char** values.

sql_variant can have a maximum length of 8016 bytes. This includes both the base type information and the base type value. The maximum length of the actual base type value is

8,000 bytes.

A **sql_variant** data type must first be cast to its base data type value before participating in operations such as addition and subtraction.

sql_variant can be assigned a default value. This data type can also have NULL as its underlying value, but the NULL values will not have an associated base type. Also, **sql_variant** cannot have another **sql_variant** as its base type.

A unique, primary, or foreign key may include columns of type **sql_variant**, but the total length of the data values that make up the key of a specific row should not be more than the maximum length of an index. This is 900 bytes.

A table can have any number of **sql_variant** columns.

sql_variant cannot be used in CONTAINSTABLE and FREETEXTTABLE.

ODBC does not fully support **sql_variant**. Therefore, queries of **sql_variant** columns are returned as binary data when you use Microsoft OLE DB Provider for ODBC (MSDASQL). For example, a **sql_variant** column that contains the character string data 'PS2091' is returned as 0x505332303931.

Comparing sql_variant Values

The **sql_variant** data type belongs to the top of the data type hierarchy list for conversion. For **sql_variant** comparisons, the SQL Server data type hierarchy order is grouped into data type families.

Data type hierarchy	Data type family
sql_variant	sql_variant
datetime2	Date and time
datetimeoffset	Date and time
datetime	Date and time
smalldatetime	Date and time
date	Date and time

Data type hierarchy	Data type family
time	Date and time
float	Approximate numeric
real	Approximate numeric
decimal	Exact numeric
money	Exact numeric
smallmoney	Exact numeric
bigint	Exact numeric
int	Exact numeric
smallint	Exact numeric
tinyint	Exact numeric
bit	Exact numeric
nvarchar	Unicode
nchar	Unicode
varchar	Unicode
char	Unicode
varbinary	Binary
binary	Binary
uniqueidentifier	Uniqueidentifier

The following rules apply to **sql_variant** comparisons:

- When **sql_variant** values of different base data types are compared and the base data types are in different data type families, the value whose data type family is higher in the hierarchy chart is considered the greater of the two values.

- When **sql_variant** values of different base data types are compared and the base data types are in the same data type family, the value whose base data type is lower in the hierarchy chart is implicitly converted to the other data type and the comparison is then made.
- When **sql_variant** values of the **char**, **varchar**, **nchar**, or **nvarchar** data types are compared, their collations are first compared based on the following criteria: LCID, LCID version, comparison flags, and sort ID. Each of these criteria are compared as integer values, and in the order listed. If all of these criteria are equal, then the actual string values are compared according to the collation.

Converting sql_variant Data

When handling the **sql_variant** data type, SQL Server supports implicit conversions of objects with other data types to the **sql_variant** type. However, SQL Server does not support implicit conversions from **sql_variant** data to an object with another data type.

Restrictions

The following table lists the types of values that cannot be stored by using **sql_variant**:

varchar(max)	**varbinary(max)**
nvarchar(max)	**xml**
text	**ntext**
image	**rowversion** (**timestamp**)
sql_variant	**geography**
hierarchyid	**geometry**
User-defined types	**datetimeoffset**[1]

[1] SQL Server 2012 and greater do not restrict **datetimeoffset**.

Examples

A. Using a sql_variant in a table

The following example, creates a table with a sql_variant data type. Then the example retrieves `SQL_VARIANT_PROPERTY` information about the `colA` value `46279.1` where `colB` `=1689`, given that `tableA` has `colA` that is of type `sql_variant` and `colB`.

SQL Copy

```
CREATE   TABLE tableA(colA sql_variant, colB int)
INSERT INTO tableA values ( cast (46279.1 as decimal(8,2)), 1689)
SELECT   SQL_VARIANT_PROPERTY(colA,'BaseType') AS 'Base Type',
         SQL_VARIANT_PROPERTY(colA,'Precision') AS 'Precision',
         SQL_VARIANT_PROPERTY(colA,'Scale') AS 'Scale'
FROM      tableA
WHERE      colB = 1689
```

Here is the result set. Note that each of these three values is a **sql_variant**.

Copy

```
Base Type    Precision    Scale
---------    ---------    -----
decimal      8           2

(1 row(s) affected)
```

B. Using a sql_variant as a variable

The following example, creates a variable using the sql_variant data type, and then retrieves `SQL_VARIANT_PROPERTY` information about a variable named @v1.

SQL Copy

```
DECLARE @v1 sql_variant;
SET @v1 = 'ABC';
SELECT @v1;
SELECT SQL_VARIANT_PROPERTY(@v1, 'BaseType');
SELECT SQL_VARIANT_PROPERTY(@v1, 'MaxLength');
```

See also

CAST and CONVERT (Transact-SQL)
SQL_VARIANT_PROPERTY (Transact-SQL)