VECTOR Data Type (Transact-SQL)

Article • 09/11/2024

Applies to: ✓ Azure SQL Database ✓ Azure SQL Managed Instance

ⓘ **Note**

The use of varbinary(8000) datatype is now deprecated in favor of the new **vector** type. Please note that this data type is currently in preview and is subject to change. Preview features are not intended for production use and are subject to additional terms of use.

> The functions **ISVECTOR**, **JSON_ARRAY_TO_VECTOR** and **VECTOR_TO_JSON_ARRAY**, which work with the varbinary(8000) datatype, will be phased out after the private preview. These functions will remain available until the end of the private preview to assist early adopters in transitioning to the new vector type.

The **VECTOR** data type is designed to store vector data optimized for operations such as similarity search and machine learning applications.

For more information on working with Vector data in SQL Database, see

- Announcing Early Adopter Preview - Native Vector Support in Azure SQL
- Intelligent applications with Azure SQL Database

Syntax

The usage syntax for the VECTOR type is similar to all other SQL Server data types in a table.

syntaxsql

```
column_name VECTOR(d) [NOT NULL | NULL]
```

In this syntax, **'d'** denotes the vector's dimensions and is a required parameter. A vector, to be valid, must have at least one dimension. At the moment, the maximum number of dimensions supported is 1998.

The **vector** type cannot be used without specifying 'd'. Dimension is a property of the vector data type and not the associated column or variable.

All column-level constraints are blocked, except for **NULL/NOT NULL** constraints

Examples

Example 1

The **vector** type can be used in column definition contained in a `CREATE TABLE` statement, for example:

The following example creates a table with a vector column and inserts data into it.

SQL

```sql
CREATE TABLE dbo.vectors
(
  id INT PRIMARY KEY,
  v VECTOR(3) NOT NULL
);

INSERT INTO dbo.vectors (id, v) VALUES (1, CAST(N'[0.1, 2, 30]' AS VECTOR(3)));
INSERT INTO dbo.vectors (id, v) VALUES (2, CONVERT(VECTOR(3), '[0.1, 2, 30]'));
```

Example 2

The following example declares vectors using the new **vector** data type and calculates distances using the VECTOR_DISTANCE function.

SQL

```sql
DECLARE @v1 VECTOR(2) = CAST(N'[1,1]' AS VECTOR(2));
DECLARE @v2 VECTOR(2) = CAST(N'[-1,-1]' AS VECTOR(2));

SELECT
    VECTOR_DISTANCE('euclidean', @v1, @v2) AS euclidean,
    VECTOR_DISTANCE('cosine', @v1, @v2) AS cosine,
    VECTOR_DISTANCE('dot', @v1, @v2) AS negative_dot_product;
```

Example 3

The following example creates a vector with three dimensions from a string with a JSON array.

SQL

```sql
DECLARE @v VECTOR(3) = CAST(N'[1, 2, 3]' AS VECTOR(3));

SELECT
    VECTOR_NORM(@v, 'norm2') AS EuclideanNorm,
    VECTOR_NORM(@v, 'norm1') AS norm1,
    VECTOR_NORM(@v, 'norminf') AS norminf;
```

The expected return values would be:

Expand table

EuclideanNorm	norm1	norminf
3.7416573867739413	6.0	3.0

Feature availability

The new native **vector** type is currently in preview for Azure SQL Database and Azure SQL Managed Instance (configured with the **Always-up-to-date** update policy).

The new **vector** type is available under all database compatibility levels.

Behavior

- All column-level constraints are blocked, except for NULL/NOT NULL constraints.
- Altering VECTOR columns to other types is not permitted.
- NULL vectors are supported:

  SQL

  ```sql
  DECLARE @v VECTOR(10) = NULL
  ```

- DEFAULT and CHECK constraints are not supported for VECTOR columns.
- Key constraints, such as PRIMARY KEY or FOREIGN KEY, are not supported for VECTOR columns. This is because equality, joins using vector columns as keys, and sort orders do not apply to VECTOR data types.

- There is no notion of uniqueness for vectors, so unique constraints are not applicable.
- Checking the range of values within a vector is also not applicable.
- Vectors do not support comparison, addition, subtraction, multiplication, division, concatenation, or any other mathematical, logical, and compound assignment operators.
- B-tree indexes and columnstore indexes are not allowed on **vector** columns.
- **vector** columns cannot be used in memory-optimized tables.

Limitations of Vector Type (Private Preview)

- The maximum dimensions supported in this release is 1998.
- The default type for dimensions is `Real`, and the only supported type for dimensions in the preview is a 4-byte float.
- The preview does not support implicit conversion from any data type to VECTOR.
- Data insertion into VECTOR type columns requires an explicit cast or convert from **varchar**, and **nvarchar** types.

> [!NOTE] Support for implicit casting will be introduced soon.

- As a limitation of the preview, due to the lack of support for implicit conversion, bcp in and Bulk Insert operations will not be supported for the VECTOR data type.
- While format files can be used to specify data as `varchar(max)` during bulk operations, explicit conversion to VECTOR is still required post-insertion.
- The behavior of `CAST ( ... AS VECTOR)` returns a **vector** type, but the sp_describe_first_result_set system stored procedure doesn't correct return the **vector** data type. Therefore, many data access clients and driver will see a **varchar** or **nvarchar** data type.

Function support

Note

The use of varbinary(8000) datatype is now deprecated in favor of the new **vector** type. Please note that this data type is currently in preview and is subject to change. Preview features are not intended for production use and are subject to additional terms of use.

> The functions **ISVECTOR**, **JSON_ARRAY_TO_VECTOR** and **VECTOR_TO_JSON_ARRAY**, which work with the varbinary(8000) datatype, will be phased out after the private preview. These functions will remain available until the end of the private preview to assist early adopters in transitioning to the new vector type.

All other Vector functions support the **vector** type

- **VECTOR_DISTANCE** : Calculates the distance between two vectors using a specified distance metric.
- **VECTOR_NORM**: Takes a vector as input and returns its norm, which is a measure of the vector's length or magnitude, in a specified norm type.
- **VECTOR_NORMALIZE**: Adjusts a vector so that its length is normalized to one.

For a complete list of **vector** functions, see Vector functions

Indexes

> **Note**
>
> The vector ANN index feature is currently under development and will be announced soon. Stay tuned for updates!

- B-tree indexes or columnstore indexes are not allowed on **vector** columns. However, a **vector** column can be specified as an included column in an index definition.

Conversion

- Explicit conversion using `CAST` or `CONVERT` from the **vector** type can be done to **varchar**, and **nvarchar** types.
- Similarly, only **varchar**, and **nvarchar** can be explicitly converted to the **vector** type.

> **Note**
>
> Support for explicit casting from **json** will be introduced soon.

- Currently all implicit conversions aren't allowed, similar to the behavior of **xml** and **json**.

ⓘ **Note**

Support for implicit casting from **varchar**, **nvarchar** and **json** will be introduced in future.

- The **vector** type can't be used with the **sql_variant** type or assigned to a **sql_variant** variable or column. This restriction similar to **varchar(max)**, **varbinary(max)**, **nvarchar(max)**, **xml**, **json** and CLR-based data types.

Compatibility

- The **vector** type can be used as a parameter or return type in a user-defined function, or the parameter of a stored procedure. The **vector** type is compatible with triggers and views.
- Creation of alias type using `CREATE TYPE` for the **vector** type isn't allowed. This is same behavior as **xml** and **json** type.
- Using `SELECT ... INTO` with the vector type will create a table with the vector type.

Related content

- Intelligent applications with Azure SQL Database
- Vector Functions (Transact SQL)
- Samples using Native Vector Search in Azure SQL

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A