Vector Index

After storing data in InterSystems IRIS in the VECTOR type, you may define a vector index (also called an approximate nearest neighbor index, or an ANN index) to improve the efficiency of searches issued against your stored vectors.

In a standard vector search, comparisons against an input vector must be made against every individual vector in the database. While this approach guarantees that your searches are completely accurate, it is computationally inefficient. A vector index leverages nearest neighbor algorithms to store the vectors in a sorted data structure that limits the number of comparison operations performed between an input vector and the stored vectors. As a result, when a search is performed, the system does not make comparisons with each stored vector, but instead uses the sorted data structure to eliminate vectors that are not close to input vector. This approach dramatically improves the performance of searches on a vector database, particularly when dealing with large amounts of high-dimensional data.

Note that queries that use a vector index currently do not support parallelization.

As with [standard indexes](http://use7420pzubiago:57772/csp/docbook/DocBook.UI.Page.cls?KEY=GSOD_indexes), the query optimizer may decide that the most efficient query plan does not use the vector index you have defined. To see if a query uses the vector index, [examine the query play](http://use7420pzubiago:57772/csp/docbook/DocBook.UI.Page.cls?KEY=GSOE_interpret) with the [EXPLAIN](http://use7420pzubiago:57772/csp/docbook/DocBook.UI.Page.cls?KEY=RSQL_explain) command.

Hierarchical Navigable Small World Index

InterSystems SQL allows you to define a Hierarchical Navigable Small World (HNSW) index, which uses the [HNSW algorithm](https://arxiv.org/abs/1603.09320) to create a vector index.

You can define an HNSW index using a [CREATE INDEX](http://use7420pzubiago:57772/csp/docbook/DocBook.UI.Page.cls?KEY=RSQL_createindex) statement. To define an HNSW index, the following requirements must be met:

* The HNSW index is defined on a VECTOR-typed field with a fixed length that is of type double or decimal.
* The table the index is defined on must have IDs that are bitmap supported.
* The table the index is defined on must use default storage.

There are three parameters you can specify when defining an HNSW index:

* **Distance** (required): The distance function used by the index, surrounded by quotes (**''**). There are two possible values: **Cosine** and **DotProduct**. This parameter is case insensitive.
* **M** (optional): The number of bi-directional links created for every new element during construction. This value should be a positive integer larger than 1; the value will fall between 2–100. Higher M values work better on datasets with high dimensionality or recall, while lower M values work better with low dimensionality or recall. The default value is 64.
* **efConstruct** (optional): The size of the dynamic list for the nearest neighbors. This value should be a positive integer that is larger than **M**. Larger **efConstrucut** values generally lead to better index quality, but longer construction time. There is a maximum value past which **efConstruct** does not improve the quality of the index. The default value is 64.

The following examples define HNSW indexes with various values in the parameters.

CREATE INDEX HNSWIndex ON TABLE Company.People (Biography)

AS HNSW(Distance='Cosine')

CREATE INDEX HNSWIndex ON TABLE Company.People (Biography)

AS HNSW(M=80, Distance='DotProduct')

CREATE INDEX HNSWIndex ON TABLE Company.People (Biography)

AS HNSW(M=72, efConstruct=100, Distance='Cosine')