

# METHODS AND APPARATUSES FOR WRITING AND SEARCHING VECTOR DATA IN VECTOR DATABASE

## TECHNICAL FIELD

[0001] One or more embodiments of this specification relate to the field of computers, and in particular, to methods and apparatuses for writing and searching vector data in a vector database.

## BACKGROUND

[0002] A vector database is a database having functions of storing and querying vector data, and the vector data stored therein can be derived from vectorization of text, voice, images, videos, etc. Compared with conventional databases, the vector database can process more unstructured data, such as images and audio. Vector data are often used in machine learning and deep learning, that is, data are usually represented in a vector form. The vector data can belong to a representation of privacy data of a user. For example, the vector data can come from vectorization of a face image or vectorization of a fingerprint image, and the face image and the fingerprint image are often considered as privacy data.

[0003] When searching vector data, the vector database focuses primarily on similarity calculation between vectors rather than on exact matching. When writing vector data, a vector index needs to be constructed. The vector index refers to a data structure established in the vector database to accelerate similarity search. In the existing technology, a solution for writing vector data into a vector database and a solution for searching vector data in a vector database often cannot consider both real-time performance and storage costs for large-scale vector data.

## SUMMARY

[0004] One or more embodiments of this specification describe methods and apparatuses for writing and searching vector data in a vector database, which can consider both real-time performance and storage costs.

[0005] According to a first aspect, a method for writing vector data into a vector database is

provided, including: receiving a write request for target vector data; adding index update information corresponding to the target vector data to a dynamic first index of a vector database in a memory table, where the first index is a memory-based real-time read-write vector graph index; when the memory table is refreshed to a disk, storing the index update information of the dynamic first index in the memory table into a distributed file system to obtain a static first index; and converting the static first index in the distributed file system into a second index, where the second index is a disk-based low-cost vector graph index.

**[0006]** In some possible implementations, the method further includes: combining multiple second indexes whose data volumes are less than a target value into one second index whose data volume is greater than or equal to the target value.

**[0007]** In a possible implementation, the first index is an HNSW index, and the second index is a DiskANN index.

**[0008]** Further, the index update information includes update of an adjacency relationship in the HNSW index and/or adjustment of a hierarchical structure of multiple hierarchical graphs.

**[0009]** Further, the converting the static first index in the distributed file system into a second index includes: obtaining a graph structure of a bottom layer in a static HNSW index in the distributed file system, and using the graph structure as a graph structure in DiskANN; obtaining each piece of vector data included in the vector database, and generating a compression vector respectively corresponding to each piece of vector data; and constructing an index file of the second index based on the graph structure and the compression vector, and storing the index file into the distributed file system.

**[0010]** According to a second aspect, a method for searching vector data in a vector database is provided, including: receiving a search request for target vector data; and performing target search based on the target vector data, so as to determine a search result, where the target search includes: separately performing vector search by using a dynamic first index stored in a memory, and a static first index and a second index stored in a distributed file system, and combining results obtained after respective search as a matching result set, where the first index is a memory-based real-time read-write vector graph index, the second index is a disk-based low-cost vector graph index, and the second index is obtained by converting a first index dumped from the memory to a disk.

**[0011]** In a possible implementation, multiple data rows are stored in the vector database, one

data row includes data columns respectively corresponding to multiple types of data, and the multiple types of data include vector data and another type of data.

**[0012]** Further, the search request further includes a filtering search criterion for another type of data; and the target search further includes: after the vector search, obtaining a filtering result set in a filter manner according to the filtering search criterion; and filtering the matching result set by using the filtering result set, and determining a search result according to a filtering result.

**[0013]** Further, the search request further includes a filtering search criterion for another type of data; and the target search further includes: before the vector search, obtaining a filtering result set in a filter manner according to the filtering search criterion; and the vector search includes performing search, by using the first index and the second index, in a sub-search range corresponding to the filtering result set to obtain the matching result set.

**[0014]** Further, the obtaining a filtering result set in a filter manner according to the filtering search criterion includes: obtaining several row numbers through searching according to at least one other type of data included in the filtering search criterion, where the row number is used to identify the data row; and determining, as the filtering result set, the several row numbers or vector data of target data columns included in several data rows corresponding to the several row numbers.

**[0015]** Further, the target search obtains several row numbers matching the target vector data, where the row number is used to identify the data row; and the determining a search result includes: obtaining data information of several target data columns in data rows corresponding to the matched several row numbers and postprocessing information thereof, where the postprocessing information includes statistical information in a search process.

**[0016]** According to a third aspect, an apparatus for writing vector data into a vector database is provided, including: a proxy server, configured to receive a write request for target vector data; a search node, configured to add index update information corresponding to the target vector data received by the proxy server to a dynamic first index of a vector database in a memory table, where the first index is a memory-based real-time read-write vector graph index; a data integration engine, configured to: when the memory table is refreshed to a disk, store the index update information of the dynamic first index in the memory table and added by the search node into a distributed file system to obtain a static first index; and a storage engine, configured to convert the static first index in the distributed file system and obtained by the data integration engine into a second index, where the second index is a disk-based low-cost vector graph index.

[0017] According to a fourth aspect, an apparatus for searching vector data in a vector database is provided, including: a proxy server, configured to receive a search request for target vector data; and a search node, configured to perform target search based on the target vector data received by the proxy server, so as to determine a search result, where the target search includes: separately performing vector search by using a dynamic first index stored in a memory, and a static first index and a second index stored in a distributed file system, and combining results obtained after respective search as a matching result set, where the first index is a memory-based real-time read-write vector graph index, the second index is a disk-based low-cost vector graph index, and the second index is obtained by converting a first index dumped from the memory to a disk.

5

[0018] According to a fifth aspect, a computer-readable storage medium is provided. The computer-readable storage medium stores a computer program, and when the computer program is executed in a computer, the computer is enabled to perform the method according to the first aspect or the second aspect.

10

[0019] According to a sixth aspect, a computing device is provided, including a storage device and a processor. The storage device stores executable code, and the processor executes the executable code to implement the method according to the first aspect or the second aspect.

15

[0020] According to the method and the apparatus for writing vector data into a vector database provided in the embodiments of this specification, first, a write request for target vector data is received; then, index update information corresponding to the target vector data is added to a dynamic first index of a vector database in a memory table, where the first index is a memory-based real-time read-write vector graph index; further, when the memory table is refreshed to a disk, the index update information of the dynamic first index in the memory table is stored into a distributed file system to obtain a static first index; and finally, the static first index in the distributed file system is converted into a second index, where the second index is a disk-based low-cost vector graph index. It can be understood from the above-mentioned description that in the embodiments of this specification, a distributed file system is used to replace local storage, and can be adapted to large-scale vector data. In addition, a composite index solution is used, and a static first index in the distributed file system is converted into a second index, so memory space is released. Subsequently, the first index can be used to service real-time incremental vector data, and the second index can be used to service historical vector data, so both real-time performance and storage costs can be considered.

20

25

30

[0021] According to the method and the apparatus for searching vector data in a vector database provided in the embodiments of this specification, first, a search request for target vector data is received; and then, target search is performed based on the target vector data, so as to determine a search result, where the target search includes: separately performing vector search by  
5 using a dynamic first index stored in a memory, and a static first index and a second index stored in a distributed file system, and combining results obtained after respective search as a matching result set, where the first index is a memory-based real-time read-write vector graph index, the second index is a disk-based low-cost vector graph index, and the second index is obtained by converting a first index dumped from the memory to a disk. It can be understood from the above-  
10 mentioned description that, in the embodiments of this specification, when searching the vector data, both the first index and the second index are used, thereby giving play to advantages of the first index and the second index, thereby enabling both real-time performance and storage costs to be considered.

## **BRIEF DESCRIPTION OF DRAWINGS**

15 [0022] To describe the technical solutions in the embodiments of the present disclosure more clearly, the following briefly describes the accompanying drawings needed for describing the embodiments. Clearly, the accompanying drawings in the following description show merely some embodiments of the present disclosure, and a person of ordinary skill in the art can still derive other drawings from these accompanying drawings without creative efforts.

20 [0023] FIG. 1 is a schematic diagram illustrating an implementation scenario of an embodiment, according to this specification;

[0024] FIG. 2 is a flowchart illustrating a method for writing vector data into a vector database, according to an embodiment;

[0025] FIG. 3 is a schematic diagram illustrating a processing process of writing vector data  
25 into a vector database, according to an embodiment;

[0026] FIG. 4 is a flowchart illustrating a method for searching vector data in a vector database, according to an embodiment;

[0027] FIG. 5 is a schematic diagram illustrating a hybrid search execution sequence, according to an embodiment;

[0028] FIG. 6 is a schematic diagram illustrating a hybrid search execution process of multiple types of data, according to an embodiment;

[0029] FIG. 7 is a schematic diagram illustrating a processing process of reading and writing vector data from or into a vector database, according to an embodiment;

5 [0030] FIG. 8 is a schematic block diagram illustrating an apparatus for writing vector data into a vector database, according to an embodiment; and

[0031] FIG. 9 is a schematic block diagram illustrating an apparatus for searching vector data in a vector database, according to an embodiment.

## DESCRIPTION OF EMBODIMENTS

10 [0032] The following describes the solutions provided in this specification with reference to the accompanying drawings.

[0033] FIG. 1 is a schematic diagram illustrating an implementation scenario of an embodiment, according to this specification. The implementation scenario involves writing vector data into a vector database and searching vector data in the vector database. It can be understood  
15 that large-scale vector data are stored in the vector database, for example, the scale of which can reach a level of hundreds of billions of vectors. When one piece of vector data is written into the vector database, an index of the vector database needs to be updated, so as to subsequently search the vector data based on the index. A specific used index manner directly affects performance of the vector database.

20 [0034] A common index manner is a memory-based real-time read-write vector graph index, and typically is an HNSW index. Another common index manner is a disk-based low-cost vector graph index, and typically is a DiskANN index. The two index manners have advantages and disadvantages. The HNSW index can basically use only a full-memory storage manner. Once data are written to a disk, performance thereof is basically unacceptable because multiple rounds of  
25 random read/write are involved in a retrieval process. The HNSW index has good flexibility in index construction, but is only applicable to a scenario with a small quantity of data, and storage costs are high in a massive data scenario. The DiskANN index is applicable to a scenario with large-scale data, and storage costs are low. Unlike the conventional single index manner, in the embodiments of this specification, a composite index solution is used, so both real-time

performance and storage costs can be considered.

[0035] Referring to FIG. 1, a schematic diagram of the HNSW index is shown. In some embodiments of this specification, a process of searching vector data based on an index is briefly described by using the HNSW index as an example. The HNSW index is a proximity graph, where two vertices are connected according to their proximity, and there is an edge between two vertices that are relatively close to each other. An approximate distance is generally defined by using a Euclidean distance. In the process of searching vector data, a hierarchical graph of the HNSW index is used, the top layer is an entry point and includes only a relatively long edge. As moving to a lower layer, an edge length becomes shorter and shorter, and edges become more and more. In the search process, the top layer is entered first. Near neighbors of a current point are traversed in each layer, a vertex closest to a query vector is greedily moved to until a local optimal solution of a current layer is found. In this case, a current vertex in a lower layer is transferred to, and search starts again. This process is repeated until a local optimal solution of the bottom layer is found. For example, the top layer is the second layer. Starting from an entry point of the second layer, a vertex corresponding to a local optimal solution of the second layer, a current vertex in the first layer, a vertex corresponding to a local optimal solution of the first layer, a current vertex in the 0th layer, and a vertex (where the vertex is a neighbor closest to the query vector) corresponding to a local optimal solution of the 0th layer are successively found in a direction of an arrow, and it can be understood that the 0th layer is the bottom layer.

[0036] It is worthwhile to note that this embodiment of this specification is not limited to the index manner of the HNSW index. In FIG. 1, the process of searching vector data based on a graph index is briefly described only by using the index manner as an example.

[0037] FIG. 2 is a flowchart illustrating a method for writing vector data into a vector database, according to an embodiment. The method can be based on the implementation scenario shown in FIG. 1. As shown in FIG. 2, the method for writing vector data into a vector database in some embodiments includes the following steps: Step 21: Receive a write request for target vector data; Step 22: Add index update information corresponding to the target vector data to a first index of a vector database in a memory table, where the first index is a memory-based real-time read-write vector graph index; Step 23: When the memory table is refreshed to a disk, store the index update information of the first index in the memory table into a distributed file system to obtain an updated first index; and Step 24: Convert the updated first index in the distributed file system into a second

index, where the second index is a disk-based low-cost vector graph index. The following describes specific execution manners of the above-mentioned steps.

[0038] First, in step 21, a write request for target vector data is received. It can be understood that vector data in the vector database can be written in batches, that is, before the target vector data are written, multiple pieces of vector data may already exist in the vector database.

[0039] For example, a total of 1000 pieces of vector data  $n_1$  to  $n_{1000}$  have been stored in the vector database, and a write request for vector data  $n_{1001}$  is further received.

[0040] Then, in step 22, index update information corresponding to the target vector data is added to a dynamic first index of the vector database in a memory table, where the first index is a memory-based real-time read-write vector graph index. It can be understood that the first index can be established and updated step by step. When new vector data need to be added to an existing data set, index update information can be added to the first index.

[0041] In an example, the first index is an HNSW index. Optionally, the first index can be an NSG index or an NSSG index.

[0042] Further, the index update information includes update of an adjacency relationship in the HNSW index and/or adjustment of a hierarchical structure of multiple hierarchical graphs.

[0043] Next, in step 23, when the memory table is refreshed to a disk, the index update information of the dynamic first index in the memory table is stored into a distributed file system to obtain a static first index. It can be understood that due to a limitation on a storage capacity of the memory table, after a size of the memory table exceeds a predetermined value, the memory table can be refreshed to a disk, so storage is persisted.

[0044] In some embodiments of this specification, the distributed file system is used to replace local storage and can be adapted to large-scale vector data.

[0045] Finally, in step 24, the static first index in the distributed file system is converted into a second index, where the second index is a disk-based low-cost vector graph index. It can be understood that not all first indexes are converted into second indexes. For real-time incremental vector data, a dynamic first index is further reserved, so as to implement hybrid indexes.

[0046] In some examples, the method further includes: combining multiple second indexes whose data volumes are less than a target value into one second index whose data volume is greater than or equal to the target value.

[0047] In this example, when the first index is refreshed to the distributed file system, this part

of first index is quickly converted into a second index, and when there are multiple second indexes whose data volumes are less than a target value, they are combined into one second index whose data volume is greater than or equal to the target value.

[0048] In an example, the second index is a DiskANN index. Optionally, the second index can be a SCANN index.

[0049] Further, the converting the static first index in the distributed file system into a second index includes: obtaining a graph structure of a bottom layer in a static HNSW index in the distributed file system, and using the graph structure as a graph structure in DiskANN; obtaining each piece of vector data included in the vector database, and generating a compression vector respectively corresponding to each piece of vector data; and constructing an index file of the second index based on the graph structure and the compression vector, and storing the index file into the distributed file system.

[0050] In this example, a product quantization algorithm such as PQ or OPQ can be used to compress an original vector to obtain a compression vector, where the compression vector is used for rough search, and the original vector is used for exact distance calculation.

[0051] According to the method for writing vector data into a vector database provided in the embodiments of this specification, first, a write request for target vector data is received; then, index update information corresponding to the target vector data is added to a dynamic first index of a vector database in a memory table, where the first index is a memory-based real-time read-write vector graph index; further, when the memory table is refreshed to a disk, the index update information of the dynamic first index in the memory table is stored into a distributed file system to obtain a static first index; and finally, the static first index in the distributed file system is converted into a second index, where the second index is a disk-based low-cost vector graph index. It can be understood from the above-mentioned description that in the embodiments of this specification, a distributed file system is used to replace local storage, and can be adapted to large-scale vector data. In addition, a composite index solution is used, and a static first index in the distributed file system is converted into a second index, so memory space is released. Subsequently, the first index can be used to service real-time incremental vector data, and the second index can be used to service historical vector data, so both real-time performance and storage costs can be considered.

[0052] FIG. 3 is a schematic diagram illustrating a processing process of writing vector data

into a vector database, according to an embodiment. Referring to FIG. 3, first, a query node (which can also be referred to as a search node) inserts new vector data into an HNSW index and sends the new vector data to a distributed file system, and then the distributed file system writes the new vector data into a data file. Then, an index construction node constructs a DiskANN index for the new vector data, and sends the DiskANN index to the distributed file system. Finally, the distributed file system sends the DiskANN index to the query node, and the query node replaces the HNSW index with the DiskANN index, thereby releasing memory space.

**[0053]** In some embodiments of this specification, the following second index construction procedure can be used: The query node obtains an IP address of the index construction node from a control node and sends a request including data metadata information to the index construction node, and the index construction node executes a DiskANN index construction task, queries a periodic node polling result, loads the DiskANN index, and unloads the HNSW index.

**[0054]** FIG. 4 is a flowchart illustrating a method for searching vector data in a vector database, according to an embodiment. The method can be based on the implementation scenario shown in FIG. 1 and the method for writing vector data into a vector database shown in FIG. 2. As shown in FIG. 4, the method for searching vector data in a vector database in some embodiments includes the following steps: Step 41: Receive a search request for target vector data; and Step 42: Perform target search based on the target vector data, so as to determine a search result, where the target search includes: separately performing vector search by using a dynamic first index stored in a memory, and a static first index and a second index stored in a distributed file system, and combining results obtained after respective search as a matching result set, where the first index is a memory-based real-time read-write vector graph index, the second index is a disk-based low-cost vector graph index, and the second index is obtained by converting a first index dumped from the memory to a disk. The following describes specific execution manners of the above-mentioned steps.

**[0055]** First, in step 41, a search request for target vector data is received. It can be understood that the target vector data can represent an image, a video frame, etc.

**[0056]** In an example, multiple data rows are stored in the vector database, one data row includes data columns respectively corresponding to multiple types of data, and the multiple types of data include vector data and another type of data.

**[0057]** In this example, there is an association between data columns in the same data row, and

multi-mode hybrid search can be implemented. The another type of data can include text data, scalar data, document data, etc., where the document data can be JSON data. Table 1 below shows an example of a data row and a data column.

**Table 1: Example of a data row and a data column**

| Row number | Introduction to video content (Text) | Date (Scalar) | Category (Scalar) | Video frame (Vector) |
|------------|--------------------------------------|---------------|-------------------|----------------------|
| 0          | Red cross country vehicle            | 20240101      | 1                 | xxxxxxx              |
| 1          | Running rabbit                       | 20240510      | 3                 | xxxxxxx              |

5

**[0058]** Referring to Table 1, one data row includes data columns respectively corresponding to multiple types of data, where the data column for recording an introduction to video content is corresponding to a text type, the data column for recording date information is corresponding to a scalar type, the data column for recording category information is corresponding to a scalar type, and the data column for recording video frame information is corresponding to vector data.

10

**[0059]** Then, in step 42, target search is performed based on the target vector data, so as to determine a search result, where the target search includes: separately performing vector search by using a dynamic first index stored in a memory, and a static first index and a second index stored in a distributed file system, and combining results obtained after respective search as a matching result set, where the first index is a memory-based real-time read-write vector graph index, the second index is a disk-based low-cost vector graph index, and the second index is obtained by converting a first index dumped from the memory to a disk. It can be understood that a result may be found only in the first index, or may be found only in the second index, or may be found both in the first index and in the second index.

15

**[0060]** In some embodiments of this specification, in the target search, used first indexes include two types: One type is dynamic and can be inserted in real time; and the other type is static and is waiting to be replaced with a second index.

20

**[0061]** In some embodiments of this specification, single mode search of multiple types of data can be provided. For a simple scenario, a standard online analysis, processing, and query capability is provided. In addition to a vector data retrieval capability, a text data retrieval capability can also

25

be included, and multiple recall capabilities are provided for a user.

**[0062]** In some embodiments of this specification, multi-mode hybrid search of multiple types of data can be provided, various complex search semantics and a tree-based hybrid search expression are supported, and multiple types of single-mode search can be flexibly combined, such as performing retrieval on vector data under certain categories of labels and performing vector retrieval on a result set of text retrieval, thereby providing a user with complex recall semantics.

**[0063]** Hybrid search can be performed in combination with two search manners: filtering of another type of data and vector search. A sequence of filtering of another type of data and vector search can be manually configured, or can be automatically generated. Filtering of another type of data can be performed in a predicate filtering manner or a text filtering manner.

**[0064]** Further, the search request further includes a filtering search criterion for another type of data; and the target search further includes: after the vector search, obtaining a filtering result set in a filter manner according to the filtering search criterion; and filtering the matching result set by using the filtering result set, and determining a search result according to a filtering result.

**[0065]** In this example, filtering of another type of data is after vector search, and is thus referred to as post filtering.

**[0066]** Further, the search request further includes a filtering search criterion for another type of data; and the target search further includes: before the vector search, obtaining a filtering result set in a filter manner according to the filtering search criterion; and the vector search includes performing search, by using the first index and the second index, in a sub-search range corresponding to the filtering result set to obtain the matching result set.

**[0067]** In this example, filtering of another type of data is before vector search, and is thus referred to as pre-filtering.

**[0068]** FIG. 5 is a schematic diagram illustrating a hybrid search execution sequence, according to an embodiment. Referring to FIG. 5, this embodiment of this specification implements efficient search by using a dynamically configurable hybrid search execution sequence. For example, filtering of another type of data is predicate filtering. Predicate filtering can implement filtering of structured data. In a hybrid search mode of vector and structured data, if a filtering rate of the structured data is high, filtering of the structured data is performed first, and then vector searching is performed, to avoid that recalled data when vector search is performed first do not satisfy filtering of the structured data, resulting in a loss of a quantity of recall rows. If

the filtering rate of the structured data is low, vector search is performed first, and then filtering of the structured data is performed according to a result of vector search, thereby greatly reducing a calculation amount of subsequent filtering. Search performance is optimized while ensuring the quantity of recalled rows through searching as much as possible.

5    **[0069]**     Further, the obtaining a filtering result set in a filter manner according to the filtering search criterion includes: obtaining several row numbers through searching according to at least one other type of data included in the filtering search criterion, where the row number is used to identify the data row; and determining, as the filtering result set, the several row numbers or vector data of target data columns included in several data rows corresponding to the several row numbers.

10   **[0070]**     Further, the obtaining several row numbers through searching according to at least one other type of data included in the filtering search criterion includes: obtaining a first row number set through searching according to first type of data included in the filtering search criterion; and obtaining, from the first row number set, a second row number set formed by several row numbers through searching according to second type of data included in the filtering search criterion.

15   **[0071]**     Further, the another type of data includes at least one of the following: text data, scalar data, and document data.

**[0072]**     FIG. 6 is a schematic diagram illustrating a hybrid search execution process of multiple types of data, according to an embodiment. Referring to FIG. 6, a first row number set is first obtained through searching based on scalar filtering. For example, a search criterion for scalar  
20   filtering is "category = 3389", and then a second row number set formed by several row numbers is obtained through searching based on text filtering. For example, a search criterion for text filtering is "introduction to video content includes a keyword "red apple", and then the second row number set or vector data of a target data column included in the several data rows corresponding to the second row number set is determined as a pre-filtering result set. Then, vector search is  
25   performed on the pre-filtering result set, and vector search depends on an HNSW index and a DiskANN index. A union set of search results of the two indexes is taken to obtain a union set row number, and finally a final result set is read and returned.

**[0073]**     Further, the target search obtains several row numbers matching the target vector data, where the row number is used to identify the data row; and the determining a search result includes:  
30   obtaining data information of several target data columns in data rows corresponding to the matched several row numbers and postprocessing information thereof, where the postprocessing

information includes statistical information in a search process.

[0074] The embodiments of this specification can maximize execution efficiency by performing two phases of search and inverse lookup. A search link is implemented in two phases. In the search phase, a row number is recalled according to a search criterion, and specific content data are not read. In the reverse lookup phase, content data are read and postprocessing information is obtained for the recalled row number, so invalid reading and postprocessing of intermediate recall data are avoided, and execution efficiency is maximized. The postprocessing information can include a matching score, a hit rate, etc.

[0075] According to the method for searching vector data in a vector database provided in the embodiments of this specification, first, a search request for target vector data is received; and then, target search is performed based on the target vector data, so as to determine a search result, where the target search includes: separately performing vector search by using a dynamic first index stored in a memory, and a static first index and a second index stored in a distributed file system, and combining results obtained after respective search as a matching result set, where the first index is a memory-based real-time read-write vector graph index, the second index is a disk-based low-cost vector graph index, and the second index is obtained by converting a first index dumped from the memory to a disk. It can be understood from the above-mentioned description that, in the embodiments of this specification, when searching the vector data, both the first index and the second index are used, thereby giving play to advantages of the first index and the second index, thereby enabling both real-time performance and storage costs to be considered.

[0076] FIG. 7 is a schematic diagram illustrating a processing process of reading and writing vector data from or into a vector database, according to an embodiment. Referring to FIG. 7, first, after receiving a write request, a write node inserts new vector data into a memory table. When the memory table is refreshed to a hard disk, the new vector data and an HNSW index are refreshed to a distributed file system, then an index construction node is triggered to construct a DiskANN index for the new vector data, and the DiskANN index is stored into the distributed file system. A query node switches from the HNSW index to the DiskANN index. After the switching is completed, the HNSW index is deleted, so memory space is released. The query node is configured to receive a query request and perform vector query by using hybrid indexes.

[0077] According to an embodiment of another aspect, an apparatus for writing vector data into a vector database is further provided, and the apparatus is configured to perform the method

for writing vector data into a vector database according to the embodiment of this specification. FIG. 8 is a schematic block diagram illustrating an apparatus for writing vector data into a vector database, according to an embodiment. As shown in FIG. 8, the apparatus 800 includes: a proxy server 81, configured to receive a write request for target vector data; a search node 82, configured to add index update information corresponding to the target vector data received by the proxy server 81 to a dynamic first index of a vector database in a memory table, where the first index is a memory-based real-time read-write vector graph index; a data integration engine 83, configured to: when the memory table is refreshed to a disk, store the index update information of the dynamic first index in the memory table and added by the search node 82 into a distributed file system to obtain a static first index; and a storage engine 84, configured to convert the static first index in the distributed file system and obtained by the data integration engine 83 into a second index, where the second index is a disk-based low-cost vector graph index.

[0078] Optionally, in an embodiment, the storage engine 84 is further configured to combine obtained multiple second indexes whose data volumes are less than a target value into one second index whose data volume is greater than or equal to the target value.

[0079] Optionally, in an embodiment, the first index is an HNSW index, and the second index is a DiskANN index.

[0080] Further, the index update information includes update of an adjacency relationship in the HNSW index and/or adjustment of a hierarchical structure of multiple hierarchical graphs.

[0081] Further, the storage engine 84 includes: an obtaining subunit, configured to obtain a graph structure of a bottom layer in a static HNSW index in the distributed file system, and use the graph structure as a graph structure in DiskANN; a compression subunit, configured to obtain each piece of vector data included in the vector database, and generate a compression vector respectively corresponding to each piece of vector data; and a storage subunit, configured to construct an index file of the second index based on the graph structure obtained by the obtaining subunit and the compression vector obtained by the compression subunit, and store the index file into the distributed file system.

[0082] According to the apparatus for writing vector data into a vector database provided in the embodiments of this specification, first, the proxy server 81 receives a write request for target vector data; then, the search node 82 adds index update information corresponding to the target vector data to a dynamic first index of a vector database in a memory table, where the first index

is a memory-based real-time read-write vector graph index; further, when the memory table is refreshed to a disk, the data integration engine 83 stores the index update information of the dynamic first index in the memory table into a distributed file system to obtain a static first index; and finally, the storage engine 84 converts the static first index in the distributed file system into a second index, where the second index is a disk-based low-cost vector graph index. It can be understood from the above-mentioned description that in the embodiments of this specification, a distributed file system is used to replace local storage, and can be adapted to large-scale vector data. In addition, a composite index solution is used, and a static first index in the distributed file system is converted into a second index, so memory space is released. Subsequently, the first index can be used to service real-time incremental vector data, and the second index can be used to service historical vector data, so both real-time performance and storage costs can be considered.

**[0083]** According to an embodiment of another aspect, an apparatus for searching vector data in a vector database is further provided, and the apparatus is configured to perform the method for searching vector data in a vector database according to the embodiment of this specification. FIG. 9 is a schematic block diagram illustrating an apparatus for searching vector data in a vector database, according to an embodiment. As shown in FIG. 9, the apparatus 900 includes: a proxy server 91, configured to receive a search request for target vector data; and a search node 92, configured to perform target search based on the target vector data received by the proxy server 91, so as to determine a search result, where the target search includes: separately performing vector search by using a dynamic first index stored in a memory, and a static first index and a second index stored in a distributed file system, and combining results obtained after respective search as a matching result set, where the first index is a memory-based real-time read-write vector graph index, the second index is a disk-based low-cost vector graph index, and the second index is obtained by converting a first index dumped from the memory to a disk.

**[0084]** Optionally, in an embodiment, multiple data rows are stored in the vector database, one data row includes data columns respectively corresponding to multiple types of data, and the multiple types of data include vector data and another type of data.

**[0085]** Further, the search request further includes a filtering search criterion for another type of data; and the search node 92 includes: a first filtering subunit, configured to: after the vector search, obtain a filtering result set in a filter manner according to the filtering search criterion; and a second filtering subunit, configured to filter the matching result set by using the filtering result

set obtained by the first filtering subunit, and determine a search result according to a filtering result.

[0086] Further, the search request further includes a filtering search criterion for another type of data; and the search node 92 includes: a first filtering subunit, configured to: before the vector search, obtain a filtering result set in a filter manner according to the filtering search criterion; and the vector search includes performing search, by using the first index and the second index, in a sub-search range corresponding to the filtering result set obtained by the third filtering subunit to obtain the matching result set.

[0087] Further, the obtaining a filtering result set in a filter manner according to the filtering search criterion includes: obtaining several row numbers through searching according to at least one other type of data included in the filtering search criterion, where the row number is used to identify the data row; and determining, as the filtering result set, the several row numbers or vector data of target data columns included in several data rows corresponding to the several row numbers.

[0088] Further, the obtaining several row numbers through searching according to at least one other type of data included in the filtering search criterion includes: obtaining a first row number set through searching according to first type of data included in the filtering search criterion; and obtaining, from the first row number set, a second row number set formed by several row numbers through searching according to second type of data included in the filtering search criterion.

[0089] Further, the another type of data includes at least one of the following: text data, scalar data, and document data.

[0090] Further, the target search obtains several row numbers matching the target vector data, where the row number is used to identify the data row; and the determining a search result includes: obtaining data information of several target data columns in data rows corresponding to the matched several row numbers and postprocessing information thereof, where the postprocessing information includes statistical information in a search process.

[0091] According to the apparatus for searching vector data in a vector database provided in the embodiments of this specification, first, the proxy server 91 receives a search request for target vector data; and then, the search node 92 performs target search based on the target vector data, so as to determine a search result, where the target search includes: separately performing vector search by using a dynamic first index stored in a memory, and a static first index and a second index stored in a distributed file system, and combining results obtained after respective search as

a matching result set, where the first index is a memory-based real-time read-write vector graph index, the second index is a disk-based low-cost vector graph index, and the second index is obtained by converting a first index dumped from the memory to a disk. It can be understood from the above-mentioned description that, in the embodiments of this specification, when searching  
5 the vector data, both the first index and the second index are used, thereby giving play to advantages of the first index and the second index, thereby enabling both real-time performance and storage costs to be considered.

**[0092]** According to an embodiment of another aspect, a computer-readable storage medium is further provided. The computer-readable storage medium stores a computer program, and when  
10 the computer program is executed in a computer, the computer is enabled to perform the method described with reference to FIG. 2 or FIG. 4.

**[0093]** According to an embodiment of still another aspect, a computing device is further provided, including a storage device and a processor. Executable code is stored in the storage device, and when the processor executes the executable code, the method described with reference  
15 to FIG. 2 or FIG. 4 is implemented.

**[0094]** A person skilled in the art should be aware that in the above-mentioned one or more examples, the functions described in this specification can be implemented by hardware, software, firmware, or any combination thereof. When implemented by using software, these functions can be stored in a computer-readable medium or transmitted as one or more instructions or one or more  
20 pieces of code on a computer-readable medium.

**[0095]** The above-mentioned specific implementations further describe in detail the objectives, technical solutions, and beneficial effects of this application. It should be understood that the descriptions above are merely specific implementations of this application and are not intended to limit the protection scope of this application. Any modifications, equivalent replacements, or  
25 improvements made on the basis of the technical solutions of this application shall fall within the protection scope of this application.

## CLAIMS

What is claimed is:

1. A method for writing vector data into a vector database, the method comprising:  
receiving a write request for target vector data;  
5 adding index update information corresponding to the target vector data to a dynamic first index of a vector database in a memory table, wherein the first index is a memory-based real-time read-write vector graph index;  
when the memory table is refreshed to a disk, storing the index update information of the dynamic first index in the memory table into a distributed file system to obtain a static first index;  
10 and  
converting the static first index in the distributed file system into a second index, wherein the second index is a disk-based low-cost vector graph index.
2. The method according to claim 1, wherein the method further comprises:  
combining multiple second indexes whose data volumes are less than a target value into one  
15 second index whose data volume is greater than or equal to the target value.
3. The method according to claim 1, wherein the first index is an HNSW index, and the second index is a DiskANN index.
4. The method according to claim 3, wherein the index update information comprises update of an adjacency relationship in the HNSW index and/or adjustment of a hierarchical structure of  
20 multiple hierarchical graphs.
5. The method according to claim 3, wherein the converting the static first index in the distributed file system into a second index comprises:  
obtaining a graph structure of a bottom layer in a static HNSW index in the distributed file system, and using the graph structure as a graph structure in DiskANN;  
25 obtaining each piece of vector data comprised in the vector database, and generating a compression vector respectively corresponding to each piece of vector data; and  
constructing an index file of the second index based on the graph structure and the compression vector, and storing the index file into the distributed file system.
6. A method for searching vector data in a vector database, the method comprising:  
30 receiving a search request for target vector data; and

performing target search based on the target vector data, so as to determine a search result, wherein the target search comprises: separately performing vector search by using a dynamic first index stored in a memory, and a static first index and a second index stored in a distributed file system, and combining results obtained after respective search as a matching result set, wherein  
5 the first index is a memory-based real-time read-write vector graph index, the second index is a disk-based low-cost vector graph index, and the second index is obtained by converting a first index dumped from the memory to a disk.

7. The method according to claim 6, wherein multiple data rows are stored in the vector database, one data row comprises data columns respectively corresponding to multiple types of  
10 data, and the multiple types of data comprise vector data and another type of data.

8. The method according to claim 7, wherein the search request further comprises a filtering search criterion for another type of data; and the target search further comprises:

after the vector search, obtaining a filtering result set in a filter manner according to the filtering search criterion; and

15 filtering the matching result set by using the filtering result set, and determining a search result according to a filtering result.

9. The method according to claim 7, wherein the search request further comprises a filtering search criterion for another type of data; and the target search further comprises:

before the vector search, obtaining a filtering result set in a filter manner according to the  
20 filtering search criterion; and

the vector search comprises performing search, by using the first index and the second index, in a sub-search range corresponding to the filtering result set to obtain the matching result set.

10. The method according to claim 8 or 9, wherein the obtaining a filtering result set in a filter manner according to the filtering search criterion comprises:

25 obtaining several row numbers through searching according to at least one other type of data comprised in the filtering search criterion, wherein the row number is used to identify the data row; and

determining, as the filtering result set, the several row numbers or vector data of target data columns comprised in several data rows corresponding to the several row numbers.

30 11. The method according to claim 7, wherein the target search obtains several row numbers matching the target vector data, wherein the row number is used to identify the data row; and the

determining a search result comprises: obtaining data information of several target data columns in data rows corresponding to the matched several row numbers and postprocessing information thereof, wherein the postprocessing information comprises statistical information in a search process.

- 5           12. An apparatus for writing vector data into a vector database, the apparatus comprising:  
a proxy server, configured to receive a write request for target vector data;  
a search node, configured to add index update information corresponding to the target vector data received by the proxy server to a dynamic first index of a vector database in a memory table, wherein the first index is a memory-based real-time read-write vector graph index;
- 10           a data integration engine, configured to: when the memory table is refreshed to a disk, store the index update information of the dynamic first index in the memory table and added by the search node into a distributed file system to obtain a static first index; and  
a storage engine, configured to convert the static first index in the distributed file system and obtained by the data integration engine into a second index, wherein the second index is a disk-
- 15           based low-cost vector graph index.

13. An apparatus for searching vector data in a vector database, the apparatus comprising:  
a proxy server, configured to receive a search request for target vector data; and  
a search node, configured to perform target search based on the target vector data received by the proxy server, so as to determine a search result, wherein the target search comprises: separately
- 20           performing vector search by using a dynamic first index stored in a memory, and a static first index and a second index stored in a distributed file system, and combining results obtained after respective search as a matching result set, wherein the first index is a memory-based real-time read-write vector graph index, the second index is a disk-based low-cost vector graph index, and the second index is obtained by converting a first index dumped from the memory to a disk.

- 25           14. A computer-readable storage medium, wherein the computer-readable storage medium stores a computer program, and when the computer program is executed in a computer, the computer is enabled to perform the method according to any one of claims 1 to 11.

15. A computing device, comprising a storage device and a processor, wherein the storage device stores executable code, and when executing the executable code, the processor implements
- 30           the method according to any one of claims 1 to 11.