

DATA REORGANIZATION METHOD AND APPARATUS FOR DATABASE TABLE, MEDIUM, AND COMPUTER DEVICE

TECHNICAL FIELD

[0001] The present disclosure relates to the field of database technologies, and in particular, to
5 a data reorganization method and apparatus for a database table, a medium, and a computer device.

BACKGROUND

[0002] In a database, a storage format of a database table often needs to be changed. When the
storage format is changed, data reorganization needs to be performed on the database table, that is,
data in the database table is rewritten in a new storage format. In a related technology, data
10 reorganization is generally performed in a common transaction insertion way, and overheads are
large in this way.

SUMMARY

[0003] According to a first aspect, an embodiment of the present disclosure provides a data
reorganization method for a database table. The method includes: reading data in a source table in
15 response to a DDL operation on the source table, where the DDL operation can change a storage
structure of the source table; sorting the read data, writing, in data blocks, sorted data into a disk
of a database node on which a target table is located, and recording a storage address of each data
block in the disk of the database node on which the target table is located; and after each data block
is written, generating an SSTable of the target table based on the storage address of each data block,
20 where the SSTable of the target table includes each data block.

[0004] In some embodiments, the source table and the target table are respectively database
tables in different namespaces, and a namespace in which the source table is located is isolated
from a namespace in which the target table is located; and the method further includes: after the
SSTable of the target table is generated based on the storage address of each data block, moving
25 the target table to the namespace in which the source table is located.

- 5 **[0005]** In some embodiments, each namespace includes a flag bit, and namespaces with different flag bits are isolated from each other; and the moving the target table to the namespace in which the source table is located includes: changing a name of the target table to a name of the source table; and changing a flag bit of the namespace in which the target table is located to the same flag bit as a flag bit of the namespace in which the source table is located.
- [0006]** In some embodiments, a table in the namespace in which the target table is located is in an invisible state.
- [0007]** In some embodiments, the method further includes: after moving the target table to the namespace in which the source table is located, deleting the source table.
- 10 **[0008]** In some embodiments, the target table includes a plurality of target table copies, the plurality of target table copies are located on a plurality of database nodes in a distributed database, and the plurality of database nodes include a primary node and at least one secondary node; and the writing, in data blocks, sorted data into a disk of a database node on which a target table is located includes: writing, in data blocks, the sorted data into a target table copy of the primary
15 node, and writing the data block written into the target table copy of the primary node into a redo log of the primary node; and synchronizing the redo log of the primary node to a redo log of each secondary node, so that each secondary node writes, in data blocks, the sorted data into a disk of the node based on the redo log of the node.
- [0009]** In some embodiments, the source table includes a plurality of first partitions, and
20 different first partitions are stored on different first database nodes in a distributed database; and the sorting the read data includes: separately reading data of a first partition stored in each first database node; and sorting the data of the first partition read from each first database node.
- [0010]** In some embodiments, the target table includes a plurality of second partitions, and different second partitions are stored on different second database nodes in the distributed database;
25 and the writing, in data blocks, sorted data into a disk of a database node on which a target table is located includes: writing, in data blocks based on a second partition to which the sorted data belongs, the sorted data into a disk of a second database node on which the target table is located.
- [0011]** In some embodiments, the source table includes a first target entry, and the first target entry includes constraint information and/or a foreign key of the source table; and the method
30 further includes: reading, in batches based on the SSTable of the target table, a second target entry corresponding to the first target entry in the target table, where a second target entry read each time

corresponds to at least one type of data in the target table; generating a check vector based on data corresponding to the read second target entry, where elements included in one check vector are data of a same type; and checking a check vector of data of a corresponding type in the read second target entry based on constraint information corresponding to data of each type.

5 **[0012]** In some embodiments, the source table includes a trigger; and the changing a name of the target table to a name of the source table includes: setting a write lock for the source table and the target table, and changing the name of the target table to the name of the source table after both the source table and the target table are successfully set. The method further includes: creating a trigger in the target table, where an operation of creating the trigger in the target table, an operation
10 of setting the write lock for the source table, an operation of changing the name of the target table to the name of the source table, and an operation of changing the flag bit of the namespace in which the target table is located to the same flag bit as the flag bit of the namespace in which the source table is located are included in a same transaction.

[0013] In some embodiments, the target table includes a plurality of target table copies, the
15 plurality of target table copies are located on a plurality of database nodes in a distributed database, and the plurality of database nodes include a primary node and at least one secondary node. The method further includes: creating an index of a target table copy of the primary node; writing the index of the target table copy into a redo log of the primary node; and synchronizing the redo log of the primary node to each secondary node, so that each secondary node creates an index of a
20 target table copy of the node based on a redo log of the node.

[0014] In some embodiments, the storage address of each data block in the disk of the database node on which the target table is located is recorded in a data storage structure in a memory of the database node on which the target table is located.

[0015] According to a second aspect, an embodiment of the present disclosure provides a data
25 reorganization apparatus for a database table. The apparatus includes: a reading module, configured to read data in a source table in response to a DDL operation on the source table, where the DDL operation can change a storage structure of the source table; a recording module, configured to: sort the read data, write, in data blocks, sorted data into a disk of a database node on which a target table is located, and record a storage address of each data block in the disk of
30 the database node on which the target table is located; and a generation module, configured to: after each data block is written, generate an SSTable of the target table based on the storage address

of each data block, where the SSTable of the target table includes each data block.

[0016] According to a third aspect, an embodiment of the present disclosure provides a computer-readable storage medium, where the computer-readable storage medium stores a computer program, and when the program is executed by a processor, the method according to any
5 embodiment is implemented.

[0017] According to a fourth aspect, an embodiment of the present disclosure provides a computer device, including a storage, a processor, and a computer program stored in the storage and running on the processor, where when the processor executes the program, the method according to any embodiment is implemented.

[0018] In embodiments of the present disclosure, when a storage structure of a source table changes, data in the source table is sorted in a sorting way, and sorted data is written, in data blocks, into a disk of a database node on which a target table is located. Because a storage address of each data block in the disk is recorded, after each data block is written, an SSTable of the target table can be directly generated based on the storage address of each data block, and there is no need to
15 use a common transaction way to write data into a memory, and then write the data in the memory into the disk. Therefore, overheads during data reorganization are reduced, and data reorganization efficiency is improved.

[0019] It should be understood that the above-mentioned general descriptions and the following detailed descriptions are merely used as examples and for explanation, and are not
20 intended to limit this disclosure.

BRIEF DESCRIPTION OF DRAWINGS

[0020] The accompanying drawings herein are incorporated into this specification and constitute a part of this specification. The accompanying drawings illustrate embodiments that conform to the present disclosure, and are used together with this specification to describe the
25 technical solutions of the present disclosure.

[0021] FIG. 1 is a flowchart of a data reorganization method for a database table according to an embodiment of the present disclosure;

[0022] FIG. 2 is a diagram of a data storage structure of a database node according to an embodiment of the present disclosure;

[0023] FIG. 3 is a diagram of a data storage structure of a database node according to another embodiment of the present disclosure;

[0024] FIG. 4 is a diagram of a distributed database according to an embodiment of the present disclosure;

5 [0025] FIG. 5 is a block diagram of a data reorganization apparatus for a database table according to an embodiment of the present disclosure; and

[0026] FIG. 6 is a structural diagram of a computer device according to an embodiment of the present disclosure.

DESCRIPTION OF EMBODIMENTS

10 [0027] Example embodiments are described herein in detail, and examples are shown in the accompanying drawings. When the following descriptions relate to the accompanying drawings, unless otherwise indicated, same numbers in different accompanying drawings represent same or similar elements. The implementations described in the following example embodiments do not represent all implementations consistent with this disclosure. On the contrary, the implementations
15 are merely examples of apparatuses and methods consistent with some aspects of this disclosure as detailed in the appended claims.

[0028] Terms used in this disclosure are for the purpose of describing specific embodiments and are not intended to limit this disclosure. The singular forms "one", "the" and "this" used in this disclosure and the appended claims are also intended to include plural forms unless the context
20 clearly represents other meanings. It should also be understood that the term "and/or" used in this specification refers to and includes any or all possible combinations of one or more associated and listed items. In addition, the term "at least one" in this specification represents any one of the multiple or any combination of at least two of the multiple.

[0029] It should be understood that, although terms first, second, and third may be used to
25 describe various information in this disclosure, the information should not be limited to these terms. These terms are only used to distinguish information of a same type from each other. For example, without departing from the scope of this disclosure, first information can also be referred to as second information. Similarly, second information can also be referred to as first information. Depending on the context, the term "if" used herein can be interpreted as "when" or "in response

to determining".

[0030] To make a person skilled in the art better understand the technical solutions in embodiments of the present disclosure, and make the above-mentioned objectives, features, and advantages of embodiments of the present disclosure more clear and easy to understand, the following further describes the technical solutions in embodiments of the present disclosure in detail with reference to the accompanying drawings.

[0031] In a database, a storage format of a database table often needs to be changed. A scenario in which the storage format needs to be changed includes but is not limited to receiving a data definition language (DDL) operation like modifying a primary key, modifying a column type, and modifying a character set on the database table. When the storage format is changed, data reorganization needs to be performed on the database table, that is, data in the database table is rewritten in a new storage format.

[0032] In a related technology, data reorganization is generally performed in a common transaction insertion way. In this common transaction way, processes such as recording data in a source table to a memory, obtaining a version number of a transaction, performing concurrency control, and writing the data in the memory to a disk are successively performed. The processes involve a plurality of times of compaction (compaction) of the data in the memory and the data in the disk. In addition, dependent objects are processed in rows in the common transaction way, and processing is time-consuming. In conclusion, overheads of performing data reorganization in the common transaction way are large.

[0033] Based on this, embodiments of the present disclosure provide a data reorganization method for a database table. Refer to FIG. 1. The method includes step 102 to step 106.

[0034] Step 102: Read data in a source table in response to a DDL operation on the source table, where the DDL operation can change a storage structure of the source table.

[0035] Step 104: Sort the read data, write, in data blocks, sorted data into a disk of a database node on which a target table is located, and record a storage address of each data block in the disk of the database node on which the target table is located.

[0036] Step 106: After each data block is written, generate an SSTable of the target table based on the storage address of each data block, where the SSTable of the target table includes each data block.

[0037] In embodiments of the present disclosure, when the storage structure of the source table

changes, the data in the source table is sorted in a sorting way, and the sorted data is written, in data blocks, into the disk of the database node on which the target table is located. Because the storage address of each data block in the disk is recorded, after each data block is written, the SSTable of the target table can be directly generated based on the storage address of each data block, and there is no need to use a common transaction way to write data into a memory, and then write the data in the memory into the disk. Therefore, overheads during data reorganization are reduced, and data reorganization efficiency is improved. The following specifically describes the solutions of embodiments of this disclosure.

[0038] In step 102, the source table can be any database table in a database, for example, an order table, and the order table can be used to record information such as time, an order number, and an order amount that are of a user for shopping on a network platform. In some embodiments, if the database is a distributed database, the source table can be a database table on any database node of the distributed database. In an implementation, the source table can include a plurality of partitions (referred to as first partitions), and different first partitions can be stored on different database nodes of the distributed database. The first partition included in the source table can be determined based on a DDL operation of creating the source table. If the source table includes M rows and N columns, the source table can be divided into k first partitions, and each first partition includes M/k rows and N/k columns in the source table. In addition to evenly partitioning the first partitions, different first partitions can also have unequal sizes. This is not limited in this disclosure.

[0039] A DDL operation can be performed on the source table. The DDL operation can be any operation that can change a storage structure of the source table, for example, modifying a primary key of the source table, modifying a column type of the source table, or modifying a character set of the source table. Data in the source table can be read in response to a DDL operation on the source table. When the source table is stored on a single database node, data in the source table can be read from the database node on which the source table is located. When the source table includes a plurality of first partitions, and different first partitions are stored on different database nodes, the first partition on each database node can be separately read, and data of each first partition is integrated, so that the data of the source table can be read.

[0040] In step 104, the data read from the source table can be sorted based on the DDL operation on the source table. When the source table includes a plurality of first partitions, and different first partitions are stored on different first database nodes in the distributed database, data

of a first partition stored in each first database node can be separately read, and the data of the first partition read from each first database node is sorted. The DDL operation can change a storage structure of the source table, for example, change a storage sequence of at least two rows and/or at least two columns of the source table. For example, the source table is an order table. If orders
5 in the order table are sorted in descending order of order amounts, a DDL operation can be sorting the orders in the order table in ascending order of order amounts. Alternatively, if a primary key of the order table is an order number, and fields in the order table are the order number, an order amount, and order placing time from left to right, a DDL operation can be using the order placing time in the order table as the primary key. In this case, sorted fields are the order placing time, the
10 order number, and the order amount from left to right. Alternatively, a DDL operation can be changing a quantity of first partitions included in the source table. It can be understood that the above-mentioned content is merely an example for description, and a sorting way in embodiments of the present disclosure is not limited.

[0041] After the read data is sorted, the sorted data can be divided into one or more data blocks,
15 and each data block can include at least one row of sorted data. A size of each data block can be fixed, for example, 16 K or 2 M. In some embodiments, the sorted data can be divided into one or more macro blocks, and a size of each macro block can be 2 M. Further, each macro block can be further divided into a plurality of micro blocks, and a size of each micro block can be 16 K. In this case, the sorted data can be written, in macro blocks, into the disk of the database node on which
20 the target table is located. In other words, each time, a macro block whose size is 2 M is written into the disk of the database node on which the target table is located. In some embodiments, each micro block can further be organized into a macro block after being compressed, and a quantity of micro blocks included in each macro block can be unfixed. In addition to the above-mentioned way of dividing the data block, the sorted data can be divided into data blocks in another way. For
25 example, the sorted data can be divided into a plurality of data blocks of different sizes. This is not described one by one in embodiments of the present disclosure.

[0042] Each time a data block is written, a storage address of the data block in the disk of the database node on which the target table is located is recorded. The target table can be a prebuilt database table, or can be a database table built in response to receiving the DDL operation on the
30 source table. For brevity, the disk of the database node on which the target table is located is referred to as a target disk. Each data block can be written into a continuous or discontinuous

storage address in the target disk. The recorded storage address can be an offset of the storage address of the data block in the target disk, or can be an actual storage address of the storage address of the data block in the target disk. When a size of the data block is not fixed, a size of each data block can be further recorded, and the size of the data block is associated with the storage address of the data block in the target disk.

[0043] In some embodiments, the target table includes a plurality of second partitions, and different second partitions are stored on different second database nodes in the distributed database. A quantity of second partitions included in the target table can be determined based on the DDL operation on the source table. When the target table includes a plurality of second partitions, the sorted data can be written, in data blocks based on a second partition to which the sorted data belongs, into a disk of a second database node on which the target table is located. For example, if a second partition to which a column of sorted data belongs is located on a database node A, the column of data can be written into a disk of the database node A.

[0044] In step 106, after the last data block is written, an SSTable including each data block can be generated. In some embodiments, a data storage structure of a database node can be implemented based on an LSM Tree. Refer to FIG. 2. Specifically, a data storage structure of a database node includes a memory table (memtable), a minor SSTable, and a major SSTable successively from top to bottom. The memory table is a memory data structure in the LSM Tree, and is stored in a memory of the database node. The minor SSTable is a data structure persisted in a disk in the LSM Tree, can include a plurality of macro blocks, and has a small data volume. There may be a plurality of minor SSTables. The major SSTable is a data structure persisted in the disk in the LSM Tree, is located at the bottom layer of the LSM Tree, and has a large data volume. There is generally only one major SSTable. Based on the above-mentioned data storage structure, the SSTable generated in this step can be a major SSTable of the target table. In embodiments of this disclosure, the major SSTable at the bottom layer of the LSM Tree is directly generated. Therefore, combined overheads of the LSM Tree are reduced, and overheads of the CPU and the IO are reduced.

[0045] Data stored in the database node can be queried in a top-down sequence of the data storage structure of the database node. For example, when target data is queried from a target table of a database node, the target data can be first queried from a memory table of the database node. If the target data is not queried from the memory table, the target data is then queried from a minor

SSTable of the database node. If the target data is not queried from the memory table and the minor SSTable of the database node, the target data is then queried from a major SSTable of the database node.

5 **[0046]** In some embodiments, a data block used to generate the major SSTable of the target table can be obtained through compression processing. For example, in the above-mentioned embodiment, the data block can be a macro block, and the macro block can include a plurality of compressed micro blocks. In this case, in one aspect, disk space occupied by the major SSTable can be reduced through data compression. In another aspect, a three-layer data storage structure is used, and a data query process is performed in the top-down sequence of the data storage structure
10 of the database node. Therefore, in the query process, a query is first performed in the uncompressed memory table and minor SSTable, and data needs to be queried from the major SSTable only when no related data is queried in the memory table and the minor SSTable. Therefore, even if data in the major SSTable is compressed, impact on the data query process is small.

15 **[0047]** Further, to facilitate recording a storage address, a layer of data storage structure can be added to the above-mentioned three-layer data storage structure, to record a storage address of each data block, as shown in FIG. 3. The added data storage structure can be a data storage structure in the memory. For ease of distinguishing from an original memtable in the memory, the added data storage structure can be referred to as a DDL memtable. In this way, only a storage
20 address of each data block needs to be recorded in the memory, and each data block does not need to be recorded, to reduce memory occupation.

[0048] In the above-mentioned embodiment, the data read from the source table can be inventory data in the source table, namely, data stored in the source table before the DDL operation is performed on the source table. The inventory data can finally be written into the major SSTable
25 of the target table. Further, in addition to the inventory data, the source table can further include incremental data, namely, data stored in the source table after the DDL operation is performed on the source table. The incremental data in the source table can be written into the memory table and the minor SSTable of the database node on which the target table is located, and does not need to be written into the major SSTable of each database node. Specifically, the incremental data in the
30 source table can be written into the memory table of the database node on which the target table is located, and the memory table is dumped to the minor SSTable. In this way, when compression

processing is performed on a data block included in the major SSTable, decompression processing does not need to be performed for each write. This improves data write performance.

5 **[0049]** In some embodiments, the source table and the target table are respectively database tables in different namespaces, and a namespace in which the source table is located is isolated from a namespace in which the target table is located. Optionally, a flag bit can be set for each namespace, where namespaces that have different flag bits are isolated from each other, and namespaces that have a same flag bit are not isolated from each other. A length of the flag bit can be determined based on a quantity of namespaces. For example, it is assumed that a quantity of namespaces is 2. In this case, a 1-bit flag bit can be used, for example, "0" and "1". During actual application, the namespace in which the source table is located can be a namespace visible to a user, and the namespace in which the target table is located can be a namespace invisible to the user, that is, a table in the namespace in which the target table is located is in an invisible state. A visible status of a namespace can be represented by using different flag bits. For example, the flag bit "0" indicates being invisible to the user, and the flag bit "1" indicates being visible to the user.

15 **[0050]** The source table and the target table are set in different namespaces. In one aspect, a data reorganization process can be performed without being perceived by the user; In another aspect, the target table does not occupy a name in the namespace of the source table, that is, the namespace in which the source table is located can include a database table that has a same naming way as that in the namespace in which the target table is located.

20 **[0051]** When the source table and the target table are respectively database tables in different namespaces, after step 106 is performed, the target table can further be moved to the namespace in which the source table is located. Specifically, a name of the target table can be changed to a name of the source table, or a flag bit of the namespace in which the target table is located can be changed to the same flag bit as a flag bit of the namespace in which the source table is located.

25 **[0052]** For ease of processing, when the name of the target table is changed, the name of the target table can be directly interchanged with the name of the source table. It can be understood that name interchanging is only an optional implementation. In another implementation, the name of the target table can also be changed to the name of the source table, and the name of the source table can be changed to another name, to avoid that the name of the target table is duplicated with the name of the source table. For example, it is assumed that the name of the source table is x and the name of the target table is y before name changing. In this case, the name of the target table

30

can be interchanged with the name of the source table. After the interchanging, the name of the source table is y, and the name of the target table is x. Alternatively, the name of the target table can be changed to x, and the name of the source table can be changed to z. Further, before the name of the target table is changed to the name of the source table, a write lock can be set for each of the target table and the source table. The name of the target table can be changed to the name of the source table after both the source table and the target table are successfully set.

[0053] After the target table is moved to the namespace in which the source table is located, the source table can be further deleted.

[0054] In some embodiments, the target table includes a plurality of target table copies, and the plurality of target table copies are located on a plurality of database nodes in the distributed database. Refer to FIG. 4. The plurality of database nodes in the distributed database include a primary node and at least one secondary node. In the figure, a distributed database including three database nodes is used as an example for description, where the primary node is Node 2, and the secondary node is Node 1 and Node 3. It can be understood that the above-mentioned content is merely an example description, and in another embodiment, a quantity of database nodes in a distributed database is not limited thereto.

[0055] For purposes such as redundant backups, the plurality of database nodes can include the same target table copy. When the sorted data is written, in data blocks, into the disk of the database node on which the target table is located, the sorted data needs to be separately synchronized, in data blocks, to a disk of a database node on which each target table copy is located, to maintain consistency of the target table copies.

[0056] In some embodiments, the sorted data can be written, in data blocks, into a target table copy of the primary node, and the data block written into the target table copy of the primary node is written into a redo log (redo log) of the primary node. Then, the redo log of the primary node is synchronized to a redo log of each secondary node, so that each secondary node writes, in data blocks, the sorted data into a disk of the node based on the redo log of the node.

[0057] The redo log is used to record each write operation of the database table. The database can build complete data by replaying the redo log. In embodiments of the present disclosure, data synchronization between a target table copy of each secondary node and the target table copy of the primary node is implemented by using the redo log, and there is no need to redo transaction logic in a synchronization process. Therefore, information such as a version number of a

transaction does not need to be recorded in the redo log. This reduces an amount of data in the redo log, and reduces bandwidth occupation in a target table copy synchronization process. In addition, data synchronization performed by using the redo log can reuse an existing transmission capability of the redo log, to improve development efficiency.

5 **[0058]** In some embodiments, the data block can be written, in micro blocks, into the redo log of the primary node, and the redo log of the primary node can be synchronized to the secondary node. When data blocks in the redo log of the secondary node accumulate to a size of a macro block, the macro block in the secondary node is written into a disk of the secondary node.

10 **[0059]** In the above-mentioned embodiment, a way of migrating the data in the source table to the target table is described. In addition, an dependent object in the source table can be migrated to the target table. The dependent object includes but is not limited to at least one of an index, constraint information, a foreign key, a trigger, etc..

15 **[0060]** When the dependent object includes an index, an index of the target table can be generated. For example, an index of the target table can be generated based on the SSTable generated in step 106. For another example, when the target table includes the incremental data in the source table, and the incremental data is written into the memory table of the database node on which the target table is located, an index of the target table can be generated based on the SSTable and the memory table of the target table.

20 **[0061]** Further, when the target table includes the plurality of target table copies, and the plurality of target table copies are located on the plurality of database nodes in the distributed database, an index of the target table copy of the primary node can be created based on the SSTable of the target table of the primary node in the plurality of database nodes; the index of the target table copy is written into the redo log of the primary node; and the redo log of the primary node is synchronized to each secondary node in the plurality of database nodes, so that each secondary
25 node creates an index of a target table copy of the node based on a redo log of the node. A way of synchronizing the index by using the redo log is similar to a way of synchronizing the target table copy by using the redo log. Details are not described herein again. The redo log used to synchronize the index can be the same as or different from the redo log used to synchronize the target table copy. This is not limited in this disclosure.

30 **[0062]** When the dependent object includes target entries such as constraint information and/or a foreign key, a second target entry corresponding to a first target entry in the target table can be

read in batches based on the SSTable of the target table, and the second target entry read each time corresponds to at least one type of data in the target table. The first target entry and the corresponding second target entry include the same field. The order table including the order number, the order amount, and the order placing time is still used as an example. The order number, the order amount, and the order placing time are data of different types. For example, if the first target entry in the source table is the order number, the second target entry in the target table is also the order number.

[0063] When second target entries are read in batches, data of the order number type can be read during first batch reading, data of the order amount type can be read during second batch reading, and data of the order placing time type can be read during third batch reading. Alternatively, data of the order number type and data of the order amount type can be read during first batch reading, and data of the order placing time type can be read during second batch reading.

[0064] Then, a check vector can be generated based on data corresponding to the read second target entry, and a check vector of data of each type in the read second target entry is checked.

Elements included in one check vector are data of a same type. For example, when data read at one time includes data of the order number type and data of the order amount type, a first check vector can be generated based on the data of the order number type, and a second check vector can be generated based on the data of the order amount type. Elements included in the first check vector are order numbers of different orders, and elements included in the second check vector are order amounts of different orders. Then, the first check vector can be checked based on check logic corresponding to the first check vector in the second target entry, and the second check vector is checked based on the check logic corresponding to the second check vector in the second target entry. For example, check logic corresponding to the order number can be that a length of the order number is equal to a preset length, and check logic corresponding to the order amount can be that a value range of the order amount is within a preset range. Then, each order number in the first check vector can be checked based on the length of the order number, and each order amount in the second check vector is checked based on the value range of the order amount. The first check vector is used as an example. If lengths of order numbers in the first check vector are all equal to the preset length, the check succeeds. If a length of any order number in the first check vector is not equal to the preset length, the check on the order number fails.

[0065] For another example, if the target table includes a foreign key, and the foreign key is a

primary key in a table referenced by the target table, a third check vector can be generated based on the foreign key of the target table, and a fourth check vector is generated based on the primary key of the table referenced by the target table. Then, the third check vector and the fourth check vector are checked based on check logic between the third check vector and the fourth check vector.

5 The check logic between the third check vector and the fourth check vector can be that a difference set between the third check vector and the fourth check vector is a null set. If the difference set between the third check vector and the fourth check vector is not a null set, it indicates that the check fails; otherwise, the check succeeds.

[0066] In the above-mentioned embodiment, after the SSTable of the target table is generated,
10 batch processing is directly performed on the second target entries of the target table. Therefore, the second target entries do not need to be processed in rows. This improves processing efficiency of the second target entry. It can be understood that the above-mentioned content is merely an example description. In another embodiment, other check logic can be used to check the second target entry, and types of check logic corresponding to data of a same type can be greater than 1.

15 **[0067]** Compared with a way of processing the dependent object in rows in a related technology, in this embodiment, constraint information is read in batches, and batch check is performed based on a type of data. In one aspect, the batch check can increase a hit rate of an instruction and a data cache by using a local principle of a computer. In another aspect, a vectorization technology can be used to improve check efficiency, thereby improving check
20 performance.

[0068] When the dependent object includes a trigger, a trigger can be created in the target table. An operation of creating a trigger in the target table and various operations performed after the sorted data is written, in data blocks, into the disk of the database node on which the target table is located can be included in a same transaction. In this way, it can be ensured that the trigger is
25 triggered on the target table only after data reorganization is completed, to avoid a performance problem caused by triggering the trigger on the target table during data reorganization.

[0069] For example, an operation of creating the trigger in the target table, an operation of setting the write lock for the source table, an operation of changing the name of the target table to the name of the source table, and an operation of changing the flag bit of the namespace in which
30 the target table is located to the same flag bit as the flag bit of the namespace in which the source table is located can be included in a same transaction. The transaction can be denoted as:

transaction 1: {setting the write lock for the source table; changing the name of the target table to the name of the source table; changing the flag bit of the namespace in which the target table is located to the same flag bit as the flag bit of the namespace in which the source table is located; and creating the trigger in the target table}.

5 **[0070]** Execution of transaction 1 can be completed only when all operations in the transaction 1 are executed. If any operation in transaction 1 fails to be executed, transaction 1 fails to be executed. In this way, the atomicity of the transaction can be ensured.

[0071] A person skilled in the art can understand that, in the above-mentioned method of a specific implementation, a writing sequence of steps does not indicate a strict execution sequence
10 and constitutes any limitation on an implementation process. A specific execution sequence of steps should be determined based on a function and possible internal logic of the steps.

[0072] Refer to FIG. 5. An embodiment of the present disclosure further provides a data reorganization apparatus for a database table. The apparatus includes: a reading module 502, configured to read data in a source table in response to a DDL operation on the source table, where
15 the DDL operation can change a storage structure of the source table; a recording module 504, configured to: sort the read data, write, in data blocks, sorted data into a disk of a database node on which a target table is located, and record a storage address of each data block in the disk of the database node on which the target table is located; and a generation module 506, configured to: after each data block is written, generate an SSTable of the target table based on the storage
20 address of each data block, where the SSTable of the target table includes each data block.

[0073] In some embodiments, functions or included modules of the apparatus provided in the embodiment of the present disclosure can be configured to execute the method described in the above-mentioned method embodiment. For a specific implementation of the apparatus, refer to the description of the above-mentioned method embodiment. For brevity, details are not described
25 herein again.

[0074] An embodiment of this specification further provides a computer device, including at least a storage, a processor, and a computer program stored in the storage and running on the processor, where when the processor executes the program, the method according to any embodiment is implemented.

30 **[0075]** FIG. 6 is a diagram of a more specific hardware structure of a computing device according to an embodiment of this specification. The device can include a processor 602, a

memory 604, an input/output interface 606, a communication interface 608, and a bus 610. The processor 602, the memory 604, the input/output interface 606, and the communication interface 608 are communicatively connected to each other through the bus 610 in the device.

5 **[0076]** The processor 602 can be implemented by using a common central processing unit (CPU), a microprocessor, an application-specific integrated circuit (ASIC), one or more integrated circuits, etc., and is configured to execute a related program, to implement the technical solutions provided in embodiments of this specification. The processor 602 can further include a graphics card, where the graphics card can be a Nvidia titan X graphics card, a 1080Ti graphics card, etc..

10 **[0077]** The memory 604 can be implemented in a form of a read-only memory (ROM), a random access memory (RAM), a static storage device, a dynamic storage device, etc.. The memory 604 can store an operating system and another service program. When the technical solutions provided in embodiments of this specification are implemented by using software or firmware, related program code is stored in the memory 604 and is invoked and executed by the processor 602.

15 **[0078]** The input/output interface 606 is configured to connect to an input/output module to implement an information input and output. The input/output module can be configured in the device as a component (not shown in the figure), or can be externally connected to the device to provide corresponding functions. Input devices can include a keyboard, a mouse, a touchscreen, a microphone, various sensors, etc.. Output devices can include a display, a speaker, a vibrator, an indicator, etc..

20 **[0079]** The communication interface 608 is configured to connect to a communication module (not shown in the figure), to implement a communicative interaction between the device and another device. The communication module can implement communication in a wired way (for example, a USB or a network cable), or can implement communication in a wireless way (for example, a mobile network, a Wi-Fi, or Bluetooth).

25 **[0080]** The bus 610 includes a channel for transmitting information between components (for example, the processor 602, the memory 604, the input/output interface 606, and the communication interface 608) of the device.

30 **[0081]** It should be noted that, although the above-mentioned device only shows the processor 602, the memory 604, the input/output interface 606, the communication interface 608, and the bus 610, during specific implementation, the device can further include another component

required for implementing normal running. In addition, a person skilled in the art can understand that the above-mentioned device can include only components required for implementing the solutions in embodiments of this specification, and does not need to include all the components shown in the figure.

5 **[0082]** An embodiment of the present disclosure further provides a computer-readable storage medium, where the computer-readable storage medium stores a computer program, and when the program is executed by a processor, the method according to any embodiment is implemented.

[0083] Computer-readable media, including permanent and non-permanent, removable and non-removable media, can be implemented by any method or technology for information storage.
10 The information can be computer-readable instructions, a data structure, a program module, or other data. Examples of the computer storage medium include but are not limited to a phase change random access memory (PRAM), a static random access memory (SRAM), a dynamic random access memory (DRAM), another type of random access memory (RAM), a read-only memory (ROM), an electrically erasable programmable read-only memory (EEPROM), a flash memory or
15 another memory technology, a compact disc read-only memory (CD-ROM), a digital versatile disc (DVD) or another optical storage, a cassette magnetic tape, a magnetic tape/magnetic disk storage, another magnetic storage device, or any other non-transmission medium. The computer storage medium can be configured to store information that can be accessed by a computing device. As specified in this specification, the computer-readable medium does not include transitory
20 computer-readable media (transitory media), such as a modulated data signal and carrier.

[0084] It can be learned from the above-mentioned description of the implementation that a person skilled in the art can clearly understand that embodiments of this specification can be implemented by using software and a required universal hardware platform. Based on such an understanding, a technical solution in embodiments of this specification essentially or a part
25 contributing to a related technology can be implemented in a form of a software product. The computer software product can be stored in a storage medium, for example, a ROM/RAM, a magnetic disk, or an optical disc, and includes several instructions for instructing a computer device (which can be a personal computer, a server, or a network device) to execute the methods described in embodiments or some parts of the embodiments of this specification.

30 **[0085]** The systems, apparatuses, modules, or units described in the above-mentioned embodiments can be specifically implemented by a computer chip or an entity, or can be

implemented by a product having a specific function. A typical implementation device is a computer. A specific form of the computer can be a personal computer, a laptop computer, a cellular phone, an image acquisition device phone, a smartphone, a personal digital assistant, a media player, a navigation device, an email sending and receiving device, a game console, a tablet
5 computer, a wearable device, or a combination of any of these devices.

[0086] The embodiments of this specification are described in a progressive way. For same or similar parts in the embodiments, refer to each other. Each embodiment focuses on a difference from other embodiments. Particularly, the apparatus embodiment is basically similar to the method embodiment, and therefore is briefly described. For a related part, refer to some descriptions in the
10 method embodiment. The above-mentioned apparatus embodiment is merely an example. The modules described as separate parts can or cannot be physically separated. When the solutions in embodiments of this specification are implemented, functions of modules can be implemented in the same or more pieces of software and/or hardware. Some or all of the modules can be selected based on an actual need to implement the objectives of the solutions in embodiments. A person of
15 ordinary skill in the art can understand and implement the solutions without paying creative efforts.

[0087] The above-mentioned descriptions are merely specific implementations of embodiments of this specification. It should be noted that a person of ordinary skill in the art can make some improvements and embellishes without departing from the principles of embodiments of this specification. The improvements and embellishes shall also fall within the protection scope
20 of embodiments of this specification.

CLAIMS

What is claimed is:

1. A data reorganization method for a database table, wherein the method comprises:

reading data in a source table in response to a DDL operation on the source table, wherein the

5 DDL operation can change a storage structure of the source table;

sorting the read data, writing, in data blocks, sorted data into a disk of a database node on which a target table is located, and recording a storage address of each data block in the disk of the database node on which the target table is located; and

after each data block is written, generating an SSTable of the target table based on the storage
10 address of each data block, wherein the SSTable of the target table comprises each data block.

2. The method according to claim 1, wherein the source table and the target table are respectively database tables in different namespaces, and a namespace in which the source table is located is isolated from a namespace in which the target table is located; and the method further comprises:

15 after the SSTable of the target table is generated based on the storage address of each data block, moving the target table to the namespace in which the source table is located.

3. The method according to claim 2, wherein each namespace comprises a flag bit, and namespaces with different flag bits are isolated from each other; and the moving the target table to the namespace in which the source table is located comprises:

20 changing a name of the target table to a name of the source table; and

changing a flag bit of the namespace in which the target table is located to the same flag bit as a flag bit of the namespace in which the source table is located.

4. The method according to claim 2, wherein a table in the namespace in which the target table is located is in an invisible state.

25 5. The method according to claim 2, wherein the method further comprises:

after moving the target table to the namespace in which the source table is located, deleting the source table.

6. The method according to claim 1, wherein the target table comprises a plurality of target table copies, the plurality of target table copies are located on a plurality of database nodes in a
30 distributed database, and the plurality of database nodes comprise a primary node and at least one

secondary node; and the writing, in data blocks, sorted data into a disk of a database node on which a target table is located comprises:

writing, in data blocks, the sorted data into a target table copy of the primary node, and writing the data block written into the target table copy of the primary node into a redo log of the primary

5 node; and

synchronizing the redo log of the primary node to a redo log of each secondary node, so that each secondary node writes, in data blocks, the sorted data into a disk of the node based on the redo log of the node.

7. The method according to claim 1, wherein the source table comprises a plurality of first
10 partitions, and different first partitions are stored on different first database nodes in a distributed database; and the sorting the read data comprises:

separately reading data of a first partition stored in each first database node; and

sorting the data of the first partition read from each first database node.

8. The method according to claim 7, wherein the target table comprises a plurality of second
15 partitions, and different second partitions are stored on different second database nodes in the distributed database; and the writing, in data blocks, sorted data into a disk of a database node on which a target table is located comprises:

writing, in data blocks based on a second partition to which the sorted data belongs, the sorted data into a disk of a second database node on which the target table is located.

9. The method according to claim 1, wherein the source table comprises a first target entry,
20 and the first target entry comprises constraint information and/or a foreign key of the source table; and the method further comprises:

reading, in batches based on the SSTable of the target table, a second target entry
corresponding to the first target entry in the target table, wherein a second target entry read each
25 time corresponds to at least one type of data in the target table;

generating a check vector based on data corresponding to the read second target entry,
wherein elements comprised in one check vector are data of a same type; and

checking a check vector of data of each type in the read second target entry.

10. The method according to claim 3, wherein the source table comprises a trigger; and the
30 changing a name of the target table to a name of the source table comprises:

setting a write lock for the source table and the target table, and changing the name of the

target table to the name of the source table after both the source table and the target table are successfully set; and

the method further comprises:

creating a trigger in the target table, wherein

5 an operation of creating the trigger in the target table, an operation of setting the write lock for the source table, an operation of changing the name of the target table to the name of the source table, and an operation of changing the flag bit of the namespace in which the target table is located to the same flag bit as the flag bit of the namespace in which the source table is located are comprised in a same transaction.

10 11. The method according to claim 1, wherein the target table comprises a plurality of target table copies, the plurality of target table copies are located on a plurality of database nodes in a distributed database, and the plurality of database nodes comprise a primary node and at least one secondary node; and the method further comprises:

creating an index of a target table copy of the primary node;

15 writing the index of the target table copy into a redo log of the primary node; and

synchronizing the redo log of the primary node to each secondary node, so that each secondary node creates an index of a target table copy of the node based on a redo log of the node.

12. The method according to claim 1, wherein the storage address of each data block in the disk of the database node on which the target table is located is recorded in a data storage structure
20 in a memory of the database node on which the target table is located.

13. A data reorganization apparatus for a database table, wherein the apparatus comprises:

a reading module, configured to read data in a source table in response to a DDL operation on the source table, wherein the DDL operation can change a storage structure of the source table;

a recording module, configured to: sort the read data, write, in data blocks, sorted data into a
25 disk of a database node on which a target table is located, and record a storage address of each data block in the disk of the database node on which the target table is located; and

a generation module, configured to: after each data block is written, generate an SSTable of the target table based on the storage address of each data block, wherein the SSTable of the target table comprises each data block.

30 14. A computer-readable storage medium, wherein the computer-readable storage medium stores a computer program, and when the program is executed by a processor, the method

according to any one of claims 1 to 12 is implemented.

15. A computer device, comprising a storage, a processor, and a computer program stored in the storage and running on the processor, wherein when the processor executes the program, the method according to any one of claims 1 to 12 is implemented.

5