DATA STORAGE METHOD AND APPARATUS, DEVICE, STORAGE MEDIUM AND PROGRAM PRODUCT

This application claims priority to Chinese Patent Application No. 202410326283.2, filed to the China National Intellectual Property Administration on March 20, 2024, and entitled "DATA STORAGE METHOD AND APPARATUS, DEVICE, STORAGE MEDIUM AND PROGRAM PRODUCT", the entire contents of which are incorporated herein by reference.

TECHNICAL FIELD

The present disclosure relates to the field of computers, and in particular to a data storage method and apparatus, a device, a storage medium and a program product.

BACKGROUND

An electronic device may include a storage device and an index tree corresponding to the storage device. The electronic device may read and write data in the storage device according to the index tree. The index tree may include a plurality of pages, and the pages may be used to index data or store data.

In the related art, the electronic device may determine, according to a data storage request, a target page to be stored in the index tree corresponding to the storage device, and store data in the target page. However, when the data to be stored in the target page is very large, the target page becomes large, which in turn requires frequent splitting operations on the target page, resulting in high resource overhead for data storage.

SUMMARY

Various aspects of the present disclosure provide a data storage method and apparatus, a device, a storage medium and a program product, so as to reduce resource overhead of data storage.

In a first aspect, an embodiment of the present disclosure provides a data storage method, including:

receiving a data storage request, where the data storage request is used to request storing first data in a disk;

when a data size of the first data is greater than or equal to a preset threshold, creating a first page corresponding to the first data;

storing the first data in the first page, and mounting the first page in an index tree corresponding to the disk.

In a possible implementation, the mounting the first page in the index tree corresponding to the disk includes:

determining a first primary page corresponding to the first page in the index tree; and

updating index-related information of the first page, updating page content of the first primary page, and updating index-related information of the first primary page, so as to mount the first page in the index tree.

In a possible implementation, the updating the index-related information of the first page includes:

determining a storage address of the first page in the disk and a first key value corresponding to the first data;

generating a first memory record corresponding to the first page, where the first memory record includes the first key value and the storage address of the first page in the disk;

storing a first page identifier of the first page in association with a memory address of the first memory record in a page mapping table corresponding to the disk; and

updating a state corresponding to the first page to a pending allocation state in the page mapping table, where the pending allocation state is used to indicate that the first page has not been mounted to the index tree.

In a possible implementation, the updating the page content of the first primary page includes:

generating a first index record based on the first key value corresponding to the first data and the first page identifier of the first page, where the first index record includes the first key value and the first page identifier; and

adding the first index record in the first primary page.

In a possible implementation, the updating the index-related information of the first primary page includes:

generating a first flush-to-disk record corresponding to the first primary page, where the first flush-to-disk record includes a memory address of the first primary page and a storage address of the first primary page in the disk; and

updating the memory address of the first primary page to a memory address of the first flush-to-disk record in the page mapping table corresponding to the disk.

In a possible implementation, the method further includes:

performing flush-to-disk processing on the first primary page in the disk based on the first flush-to-disk record;

updating, in the disk, a state corresponding to the first page in the first primary page to a confirmed allocation state, where the confirmed allocation state is used to indicate that the first page has been mounted to the index tree; and

updating the state corresponding to the first page to the confirmed allocation state in the page mapping table.

In a possible implementation, the method further includes:

receiving a data deletion request, where the data deletion request is used to request deleting second data corresponding to a second key value;

determining a second page corresponding to the second data in the index tree; and

deleting the second page and index-related information in the index tree, so as to delete the second data.

In a possible implementation, the deleting the second page and the index-related information in the index tree includes:

deleting the second page;

determining a second primary page corresponding to the second page in the index tree; and

updating page content and index-related information of the second primary page.

In a possible implementation, the deleting the second page includes:

when the second page has not been flushed to the disk, deleting the second page and a second memory record corresponding to the second page from a memory corresponding to the disk;

when the second page has been flushed to the disk, deleting the second page from both the memory and the disk, and deleting the second memory record from the memory; and

deleting a memory address of the second memory record and updating a state corresponding to the second page to a pending deletion state in the page mapping table corresponding to the disk.

In a possible implementation, the updating the page content of the second primary page includes:

determining a second index record corresponding to the second page, where the second index record includes the second key value and a second page identifier of the second page; and

updating a state corresponding to the second index record to a confirmed deletion state in the second primary page.

In a possible implementation, the updating the index-related information of the second primary page includes:

generating a second flush-to-disk record corresponding to the second primary page, where the second flush-to-disk record includes a storage address of the second primary page in the disk; and

updating a memory address of the second primary page to a memory address of the second flush-to-disk record in a page mapping table corresponding to the disk.

In a possible implementation, the method further includes:

updating a state corresponding to the second page to a confirmed deletion state in the page mapping table.

In a possible implementation, the method further includes:

receiving a data merge request, where the data merge request is used to request merging a third page and at least one update record of the third page;

generating a merge record corresponding to the third page based on the data merge request, and mounting the merge record on the third page, where the merge record is used to mount a next update record of the third page; and

performing merge processing on the third page and the at least one update record to obtain a merged page.

In a possible implementation, the mounting the merge record on the third page includes:

determining a second update record among the at least one update record, where the second update record is the last update record among the at least one update record;

adding a memory address of the second update record in the merge record; and

updating a mapping content corresponding to a third page identifier of the third page to a memory address of the merge record in a page mapping table corresponding to the disk.

In a possible implementation, after performing the merge processing on the third page and the at least one update record to obtain the merged page, the method further includes:

determining a memory address of the merged page; and

updating the memory address of the second update record to the memory address of the merged page in the merge record.

In a possible implementation, after generating the merge record corresponding to the third page based on the data merge request, the method further includes:

determining a first record count of the at least one update record; and

adding the first record count in the merge record.

In a possible implementation, after performing the merge processing on the third page and the at least one update record to obtain the merged page, the method further includes:

updating the first record count to 0 in the merge record.

In a possible implementation, the method further includes:

receiving an update request corresponding to the third page;

generating a target update record corresponding to the third page; and

adding a memory address of the merge record to the target update record, and updating the memory address of the merge record to a memory address of the target update record in a page mapping table corresponding to the disk.

In a possible implementation, the method further includes:

determining, in the disk, at least one target data block to be subjected to garbage collection, where the target data block includes a plurality of page segments;

determining a valid page segment in the at least one target data block, and storing the valid page segment in a buffer; and

deleting the at least one target data block from the disk.

In a possible implementation, the determining, in the disk, the at least one target data block to be subjected to the garbage collection includes:

determining a first data block in the disk, where the first data block is the last data block from a previous garbage collection;

determining a second data block on which a data operation is being performed in the disk;

determining a data block between the first data block and the second data block in the disk as the at least one target data block.

In a possible implementation, the determining the valid page segment in the at least one target data block includes:

determining a segment state of each page segment in the at least one target data block, where the segment state is a valid state or an invalid state; and

determining a page segment in the valid state among the at least one target data block as a valid page segment.

In a possible implementation, for any page segment, determining the segment state of the page segment in the at least one target data block includes:

determining a fourth page corresponding to the page segment, where the page segment is obtained after a flush-to-disk is performed on the fourth page;

determining a latest flush-to-disk record corresponding to the fourth page, where the latest flush-to-disk record includes a first storage address corresponding to the fourth page, and the first storage address is the earliest flush address that is currently valid for the fourth page in the disk;

when the storage address of the page segment is greater than or equal to the first storage address, determining the segment state as the valid state; and

when the storage address of the page segment is less than the first storage address, determining the segment state as the invalid state.

In a possible implementation, the method further includes:

determining a data block count in the disk;

when the data block count is greater than or equal to a preset count, storing a page mapping table corresponding to the disk;

determining a maximum page identifier in the page mapping table, a third data block that has most recently completed flushing when storage of the page mapping table begins, and a fourth data block on which a data operation is being performed when storage of the page mapping table is completed; and

generating checkpoint information based on the maximum page identifier, the third data block, and the fourth data block, and storing the checkpoint information, where the checkpoint information includes the maximum page identifier, an identifier of the third data block, and an identifier of the fourth data block.

In a possible implementation, the method further includes:

when a disk fault is determined, obtaining the page mapping table and the checkpoint information; and

performing fault recovery processing on the disk based on the page mapping table and the checkpoint information.

In a second aspect, an embodiment of the present disclosure provides a data storage apparatus, including: a receiving module, a creating module, a storage module, and a mounting module, where,

the receiving module is configured to receive a data storage request, where the data storage request is used to request storing first data in a disk;

the creating module is configured to, when a data size of the first data is greater than or equal to a preset threshold, create a first page corresponding to the first data;

the storage module is configured to store the first data in the first page; and

the mounting module is configured to mount the first page in an index tree corresponding to the disk.

In a possible implementation, the mounting module is specifically configured to:

determine a first primary page corresponding to the first page in the index tree; and

update index-related information of the first page, update page content of the first primary page, and update index-related information of the first primary page, so as to mount the first page in the index tree.

In a possible implementation, the mounting module is specifically configured to:

determine a storage address of the first page in the disk and a first key value corresponding to the first data;

generate a first memory record corresponding to the first page, where the first memory record includes the first key value and the storage address of the first page in the disk;

store a first page identifier of the first page in association with a memory address of the first memory record in a page mapping table corresponding to the disk; and

update a state corresponding to the first page to a pending allocation state in the page mapping table, where the pending allocation state is used to indicate that the first page has not been mounted to the index tree.

In a possible implementation, the mounting module is specifically configured to:

generate a first index record based on the first key value corresponding to the first data and the first page identifier of the first page, where the first index record includes the first key value and the first page identifier; and

add the first index record in the first primary page.

In a possible implementation, the mounting module is specifically configured to:

generate a first flush-to-disk record corresponding to the first primary page, where the first flush-to-disk record includes a memory address of the first primary page and a storage address of the first primary page in the disk; and

update the memory address of the first primary page to a memory address of the first flush-to-disk record in the page mapping table corresponding to the disk.

In a possible implementation, the apparatus further includes: a processing module and an updating module, where,

the processing module is configured to perform flush-to-disk processing on the first primary page in the disk based on the first flush-to-disk record;

the updating module is configured to update, in the disk, a state corresponding to the first page in the first primary page to a confirmed allocation state, where the confirmed allocation state is used to indicate that the first page has been mounted to the index tree; and

the updating module is configured to update the state corresponding to the first page to the confirmed allocation state in the page mapping table.

In a possible implementation, the apparatus further includes: a first determining module and a deleting module, where,

the receiving module is further configured to receive a data deletion request, where the data deletion request is used to request deleting second data corresponding to a second key value;

the first determining module is configured to determine a second page corresponding to the second data in the index tree; and

the deleting module is configured to delete the second page and index-related information in the index tree, so as to delete the second data.

In a possible implementation, the deleting module is specifically configured to:

delete the second page;

determine a second primary page corresponding to the second page in the index tree; and

update page content and index-related information of the second primary page.

In a possible implementation, the deleting module is specifically configured to:

when the second page has not been flushed to the disk, delete the second page and a second memory record corresponding to the second page from a memory corresponding to the disk;

when the second page has been flushed to the disk, delete the second page from both the memory and the disk, and delete the second memory record from the memory; and

delete a memory address of the second memory record and update a state corresponding to the second page to a pending deletion state in the page mapping table corresponding to the disk.

In a possible implementation, the deleting module is specifically configured to:

determine a second index record corresponding to the second page, where the second index record includes the second key value and a second page identifier of the second page; and

update a state corresponding to the second index record to a confirmed deletion state in the second primary page.

In a possible implementation, the deleting module is specifically configured to:

generate a second flush-to-disk record corresponding to the second primary page, where the second flush-to-disk record includes a storage address of the second primary page in the disk; and

update a memory address of the second primary page to a memory address of the second flush-to-disk record in a page mapping table corresponding to the disk.

In a possible implementation, the updating module is further configured to:

update a state corresponding to the second page to a confirmed deletion state in the page mapping table.

In a possible implementation, the apparatus further includes: a first generating module and a merge module, where,

the receiving module is further configured to receive a data merge request, where the data merge request is used to request merging a third page and at least one update record of the third page;

the first generating module is configured to generate a merge record corresponding to the third page based on the data merge request;

the mounting module is further configured to mount the merge record on the third page, where the merge record is used to mount a next update record of the third page; and

the merge module is configured to perform merge processing on the third page and the at least one update record to obtain a merged page.

In a possible implementation, the mounting module is specifically configured to:

determine a second update record among the at least one update record, where the second update record is the last update record among the at least one update record;

add a memory address of the second update record in the merge record; and

update a mapping content corresponding to a third page identifier of the third page to a memory address of the merge record in a page mapping table corresponding to the disk.

In a possible implementation,

the first determining module is further configured to determine a memory address of the merged page; and

the updating module is further configured to update the memory address of the second update record to the memory address of the merged page in the merge record.

In a possible implementation, the first generating module is further configured to:

determine a first record count of the at least one update record; and

add the first record count in the merge record.

In a possible implementation, the updating module is further configured to:

update the first record count to 0 in the merge record.

In a possible implementation,

the receiving module is further configured to receive an update request corresponding to the third page;

the first generating module is further configured to generate a target update record corresponding to the third page; and

the updating module is further configured to add a memory address of the merge record to the target update record, and update the memory address of the merge record to a memory address of the target update record in a page mapping table corresponding to the disk.

In a possible implementation, the apparatus further includes: a second determining module,

the second determining module is configured to determine, in the disk, at least one target data block to be subjected to garbage collection, where the target data block includes a plurality of page segments;

the second determining module is configured to determine a valid page segment in the at least one target data block;

the storage module is configured to store the valid page segment in a buffer; and

the deleting module is further configured to delete the at least one target data block from the disk.

In a possible implementation, the second determining module is specifically configured to:

determine a first data block in the disk, where the first data block is the last data block from a previous garbage collection;

determine a second data block on which a data operation is being performed in the disk;

determine a data block between the first data block and the second data block in the disk as the at least one target data block.

In a possible implementation, the second determining module is specifically configured to:

determine a segment state of each page segment in the at least one target data block, where the segment state is a valid state or an invalid state; and

determine a page segment in the valid state among the at least one target data block as a valid page segment.

In a possible implementation, for any page segment, the second determining module is specifically configured to:

determine a fourth page corresponding to the page segment, where the page segment is obtained after a flush-to-disk is performed on the fourth page;

determine a latest flush-to-disk record corresponding to the fourth page, where the latest flush-to-disk record includes a first storage address corresponding to the fourth page, and the first storage address is the earliest flush address that is currently valid for the fourth page in the disk;

when the storage address of the page segment is greater than or equal to the first storage address, determine the segment state as the valid state; and

when the storage address of the page segment is less than the first storage address, determine the segment state as the invalid state.

In a possible implementation, the apparatus further includes: a third determining module and a second generating module, where,

the third determining module is configured to determine a data block count in the disk;

the storage module is further configured to, when the data block count is greater than or equal to a preset count, store a page mapping table corresponding to the disk;

the third determining module is further configured to determine a maximum page identifier in the page mapping table, a third data block that has most recently completed flushing when storage of the page mapping table begins, and a fourth data block on which a data operation is being performed when storage of the page mapping table is completed; and

the second generating module is configured to generate checkpoint information based on the maximum page identifier, the third data block, and the fourth data block, and

store the checkpoint information, where the checkpoint information includes the maximum page identifier, an identifier of the third data block, and an identifier of the fourth data block.

In a possible implementation, the apparatus further includes: an obtaining module and a recovery module, where,

the obtaining module is configured to, when a disk fault is determined, obtain the page mapping table and the checkpoint information; and

the recovery module is configured to perform fault recovery processing on the disk based on the page mapping table and the checkpoint information.

In a third aspect, an embodiment of the present disclosure provides an electronic device, including: a memory and a processor;

the memory stores computer-executable instructions; and

the processor executes the computer-executable instructions stored in the memory, to cause the electronic device to perform the method according to any one of the first aspect.

In a fourth aspect, an embodiment of the present disclosure provides a computer-readable storage medium, where computer-executable instructions are stored in the computer-readable storage medium, and when the computer-executable instructions are executed by a processor, the method according to any one of the first aspect is implemented.

In a fifth aspect, an embodiment of the present disclosure provides a computer program product, including a computer program which, when executed by a processor, implements the method according to any one of the first aspect.

Embodiments of the present disclosure provide a data storage method and apparatus, a device, a storage medium and a program product. An electronic device may receive a data storage request. When a data size of first data is greater than or equal to a preset threshold, the electronic device may create a first page corresponding to the first data, store the first data in the first page, and mount the first page in an index tree corresponding to a disk. Since the electronic device may store the first data separately in the first page and then mount the first page in the index tree, there is no need to store the first data in a target page in the index tree, thereby avoiding frequent splitting operations on the target page caused by the first data being very large, and reducing the resource overhead of data storage.

BRIEF DESCRIPTION OF DRAWINGS

The drawings described herein are used to provide further understanding of the present disclosure and constitute a part of the present disclosure. The exemplary embodiments of the present disclosure and the descriptions thereof are used to explain the present disclosure and do not constitute improper limitations on the present disclosure.

FIG. 1 is a schematic diagram of a scenario according to an exemplary embodiment of the present disclosure.

FIG. 2 is a schematic diagram of a working principle of a Bw^e-tree according to an exemplary embodiment of the present disclosure.

FIG. 3 is a schematic flowchart of a data storage method according to an exemplary embodiment of the present disclosure.

FIG. 4 is a schematic flowchart of another data storage process according to an exemplary embodiment of the present disclosure.

FIG. 5 is a schematic diagram of a process of a data storage method according to an exemplary embodiment of the present disclosure.

FIG. 6 is a schematic flowchart of a data deletion method according to an exemplary embodiment of the present disclosure.

FIG. 7 is a schematic flowchart of another data deletion method according to an exemplary embodiment of the present disclosure.

FIG. 8 is a schematic diagram of a process of a data deletion method according to an exemplary embodiment of the present disclosure.

FIG. 9 is a schematic flowchart of an update linking mechanism according to an exemplary embodiment of the present disclosure.

FIG. 10 is a schematic diagram of a process of an update linking mechanism according to an exemplary embodiment of the present disclosure.

FIG. 11 is a schematic flowchart of a garbage collection method according to an exemplary embodiment of the present disclosure.

FIG. 12 is a schematic flowchart of a data recovery method according to an exemplary embodiment of the present disclosure.

FIG. 13 is a schematic diagram of a page lifecycle according to an exemplary embodiment of the present disclosure.

FIG. 14 is a first schematic structural diagram of a data storage apparatus according to an exemplary embodiment of the present disclosure.

FIG. 15 is a second schematic structural diagram of a data storage apparatus according to an exemplary embodiment of the present disclosure.

FIG. 16 is a schematic structural diagram of an electronic device according to an exemplary embodiment of the present disclosure.

DESCRIPTION OF EMBODIMENTS

It should be noted that the user information (including but not limited to user device information, user personal information, etc.) and data (including but not limited to data used for analysis, stored data, displayed data, etc.) involved in the present disclosure are all information and data that have been authorized by the user or have been fully authorized by all parties, and the collection, use, and processing of the relevant data need to comply with relevant laws, regulations, and standards, and corresponding operation entries are provided for the user to choose to authorize or refuse.

In order to make the objectives, technical solutions, and advantages of the present disclosure clearer, the technical solutions of the present disclosure will be clearly and completely described below in conjunction with specific embodiments of the present disclosure and the corresponding drawings. Obviously, the described embodiments are only a part of the embodiments of the present disclosure, rather than all of the embodiments. Based on the embodiments of the present disclosure, all other embodiments obtained by those of ordinary skill in the art without creative effort fall within the protection scope of the present disclosure.

FIG. 1 is a schematic diagram of a scenario provided by an exemplary embodiment of the present disclosure. Referring to FIG. 1, an electronic device may include an index tree.

The index tree may include a plurality of pages. For example, the index tree may include a page 1, and the page 1 may have corresponding page 2, ..., page n (n is an integer greater

than or equal to 1) therebelow; the page 2 may have corresponding page 2-1, page 2-2, and page 2-3 therebelow; ...; and a page n may have corresponding page n-1, and page n-2 therebelow.

When the size of the first data is greater than or equal to a preset threshold, the electronic device may store, according to a data storage request, the first data in a first page and mount the first page to the index tree.

In the related art, the electronic device may determine, according to a data storage request, a target page to be stored in an index tree corresponding to a storage device, and store data in the target page. However, when the data to be stored in the target page is very large, the target page becomes very large, which in turn requires frequent splitting operations on the target page, resulting in high resource overhead for data storage.

In an embodiment of the present disclosure, when the size of the first data is greater than the preset threshold, the electronic device may create a first page corresponding to the first data. The electronic device may store the first data in the first page, and then mount the first page to the index tree. Since the electronic device may store the first data separately in the first page and then mount the first page to the index tree, there is no need to store the first data in a target page in the index tree, thereby avoiding frequent splitting operations caused by the first data being very large in the target page, and reducing the resource overhead for data storage.

In the technical solution of the present disclosure, an index tree named Bw^{e}-tree is provided. In order to facilitate understanding of the technical solutions of the present disclosure, a working principle of the index tree Bw^{e}-tree is first explained below with reference to FIG. 2.

FIG. 2 is a schematic diagram of a working principle of a Bw^{e}-tree provided by an exemplary embodiment of the present disclosure. Referring to FIG. 2, the electronic device may include a disk and a memory.

A page mapping table may be provided in the memory, and the page mapping table may be used to manage the index tree.

The page mapping table may include page identifiers and mapping entries. For example, in the page mapping table, 0, 1, 2, ..., 9 are page identifiers, and each page identifier has a corresponding mapping entry. A mapping entry may store a corresponding memory address or storage address. Where the memory address refers to an address in the memory, and the storage address refers to an address in the disk.

Any page (also referred to as a node) in the Bw^e-tree may be identified by P. For example, a page 2 may be represented as P2.

It should be noted that different pages corresponding to the same page identifier may be distinguished by a key value range. For example, the page corresponding to page identifier 2 may be a page P2[400,500), and after updating, the page corresponding to the page identifier 2 may be a page P2[400,700).

The Bw^e-tree may include a plurality of index pages and a plurality of leaf pages. For example, in the Bw^e-tree, a page P3 and a page P8 may be index pages, and a page P2 and page P4 may be leaf pages. The page P3 may be the parent page of the page P2 and the page P4.

Each index page may have a corresponding key value range. The index page may store index records corresponding to a plurality of child pages, pointers pointing to each child page, and a pointer pointing to the right sibling page. Optionally, a pointer may be a memory address or a storage address of the corresponding page.

For example, as shown in FIG. 2, the key value range corresponding to the page P3 may be (-∞,1000), and the page P3 may store: the index record (P1,400) corresponding to the page P1, the index record (P2,700) corresponding to the page P2, the index record (P4,1000) corresponding to the page P4, and a pointer to the right sibling page P8.

Each leaf page has a corresponding key value range [low key, high key). The leaf page stores a plurality of key-value pairs within the key value range, as well as the pointer to the right sibling page. Each value has a corresponding log sequence number (Log Sequence Number, LSN).

For example, in the Bw^e-tree, the key value range of the page P2 may be [400,700), and the page P2 stores: values corresponding to keys from 400 to 700, respectively, and a pointer

to the right sibling page P4. For example, key 400 may have corresponding value1-LSN1 and value2-LSN2. Value1 is the historical data corresponding to key value 400, and value2 is the current data corresponding to key value 400.

The electronic device may perform page processing on pages in the index tree through a structural modification operation (structural modification operation, SMO).

When an operation is performed on a page in the Bw^e-tree, an operation record corresponding to the page may be generated and linked to the page, rather than directly modifying the page. Where the operation record may include a data-delta record (data-delta), a flush-to-disk record (flush-delta), and a stub record (stub-delta), etc.

Any operation record may include the memory address of the corresponding page or the memory address of the preceding operation record, a data-delta count, a page segment count, and a page size. If the operation record is a flush-to-disk record, the flush-to-disk record may further include the storage address of the page segment in the disk. Where the page size refers to the total size of the plurality of key-value pairs and LSNs in the page, i.e., the size of the logical page.

For any page, each time an operation record corresponding to the page is generated, in the mapping entry corresponding to the page, the content in the mapping entry may be updated to the memory address of the operation record, so that the mapping entry corresponding to the page points to the operation record of the page.

For example, as shown in FIG. 2, when a flush operation is performed on the page P2, i.e., the page P2 is stored as a P2 page segment 1 to the disk, a flush-to-disk record a corresponding to the page P2 may be generated. The flush-to-disk record a may include the memory address of the page P2 in the memory, a data-delta count of 0, a page segment count of 1, a page size of 4 KB (kilobytes), and the storage address 1 of the P2 page segment 1 in the disk. In the page mapping table, the memory address of the page P2 in the memory may be updated to the memory address of the flush-to-disk record a in the mapping entry corresponding to the page identifier 2.

When new data is added to the page P2, a data-delta record 1 corresponding to the page P2 may be generated. The data-delta record 1 may include: the new data added to the page P2,

i.e., a value corresponding to a key value 515, etc., the memory address of the flush-to-disk record a, a data-delta count of 1, a page segment count of 2, and a page size of 6 KB. In the page mapping table, the memory address of the flush-to-disk record a may be updated to the memory address of the data-delta record 1 in the mapping entry corresponding to the page identifier 2.

Similarly, when the value corresponding to key value 455 of the page P2 is updated, a data-delta record 2 may be generated. The data-delta record 2 may include: the new data added to the page P2, i.e., the value corresponding to the key value 455, etc., the memory address of the data-delta record 1, the data-delta count of 2, a page segment count of 2, and a page size of 7 KB. In the page mapping table, the memory address of the data-delta record 1 may be updated to the memory address of the data-delta record 2 in the mapping entry corresponding to the page identifier 2.

When a flush operation is performed on the page P2 again, i.e., the new data of the page P2 is stored as the P2 page segment 2 to the disk, a flush-to-disk record b corresponding to the page P2 may be generated. The flush-to-disk record b may include the memory address of the data-delta record 2, a data-delta count of 2, a page segment count of 2, a page size of 7 KB, and the storage address 2 of the P2 page segment 2 in the disk. In the page mapping table, the memory address of the data-delta record 2 may be updated to the memory address of the flush-to-disk record b in the mapping entry corresponding to the page identifier 2.

It should be noted that in the disk, the P2 page segment 2 may include the storage address of the P2 page segment 1. When a flush operation is performed on the page P2 again, incremental storage may be performed, i.e., only the incremental data (515...) and (455...) for the page P2 are stored, which is beneficial for reducing storage write amplification.

When the memory is insufficient, pages in the memory may be stored to the disk, the page is deleted from the memory, and a stub record corresponding to the page is stored in the memory to reduce the memory space occupied. For example, when the memory is insufficient, the page P4 may be stored to the disk, and a stub record 1 corresponding to the page P4 is stored in the memory, where the storage record 1 may include the storage address of the page P4 in the disk. In the page mapping table, the memory address of the page P4 may

be updated to the memory address of the stub record 1 in the mapping entry corresponding to a page identifier 4.

For any page, if the page is deleted, the page identifier corresponding to the page may be collected and linked together with identifiers of other free pages.

For example, as shown in FIG. 2, if a page 0, a page 9, a page 7, and a page 6 are free pages, then a page identifier 0 may point to a page identifier 9, the page identifier 9 may point to a page identifier 7, the page identifier 7 may point to a page identifier 6, and the page identifier 6 may point to the page identifier 0, so as to link the page identifiers of the four free pages together.

It should be noted that a flush operation being performed on any page is specifically as follows:

if the page is linked with a plurality of operation records, when the data-delta count in the operation records reaches a threshold, the page may be updated according to the plurality of operation records to obtain a new page, and the new page is stored as a page segment in a buffer; and

if the data-delta count in the operation records has not reached the threshold and flushing is still needed, the page and the plurality of operation records may be stored as a page segment in the buffer.

The buffer may store page segments corresponding to a plurality of pages. When the buffer is full, the plurality of page segments in the buffer may be encapsulated in a data block, a corresponding logical block number is allocated to the data block, and the data block is stored to the disk, i.e., flushed to disk. Where the logical block number refers to a sequence number corresponding to the data block. Any data block may have a corresponding logical block number. Logical block numbers are continuously and monotonically increasing. For example, in the disk, the logical block number of the first stored data block may be 001, and the logical block number of the second stored data block is 002.

It should be noted that if the logical block number of the latest data block in the current disk is 1000, it does not mean that there are 1000 data blocks in the current disk, the reason is that, in the disk, some data blocks before the latest data block with a logical block number

1000 may have already been deleted. For example, the logical block number of the latest data block in the current disk is 1000, but the 100 data blocks with logical block numbers from 1 to 100 have been deleted, then the current disk may contain 900 data blocks, and the logical block number of the latest data block is 1000.

The technical solutions shown in the present disclosure are described in detail below through specific embodiments. It should be noted that the following embodiments may exist independently or may be combined with each other, and the same or similar contents are not repeated in different embodiments.

FIG. 3 is a schematic flowchart of a data storage method provided by an exemplary embodiment of the present disclosure. Referring to FIG. 3, the method may include the following.

Step S301: Receive a data storage request.

An executive entity of the embodiment of the present disclosure may be an electronic device, or may be a data storage apparatus provided in the electronic device. The data storage apparatus may be implemented by software, or may be implemented by a combination of software and hardware. The data storage apparatus may be a processor in the electronic device. For ease of understanding, the following description takes the executive entity as the electronic device as an example.

The electronic device may include a disk. The data storage request may be used to request storing the first data in the disk.

The data storage request may include a first key value and the first data corresponding to the first key value. Optionally, the data storage request may further include an LSN number corresponding to the first data.

For example, the electronic device may receive a data storage request 1, and the data storage request 1 may include a first key value 910, data 1, and LSN1. The data storage request 1 may be used to request the electronic device to store data 1 in the disk.

Step S302: When a data size of a first data is greater than or equal to a preset threshold, create a first page corresponding to the first data.

The preset threshold may be preset manually. For example, the preset threshold may be 4K.

Since the data storage request includes the first data, the electronic device may obtain the first data after obtaining the data storage request. The electronic device may determine whether the data size of the first data is greater than or equal to the preset threshold. If so, the electronic device may create a first page corresponding to the first data in the memory. The first page may be a page outside the index tree.

For example, if the data storage request 1 includes data 1 and the size of the data 1 is 8K, the electronic device may create the first page, and the first page is assumed to be the page P30.

It should be noted that if the data size of the first data is less than the preset threshold, the electronic device may determine a target page in the index tree corresponding to the disk, and store the first data in the target page. For example, if the data size of the first data is 1K, the electronic device may determine a target page in the index tree, and the target page is assumed to be the page P2, then the electronic device may store the first data in the page P2.

Step S303: Store the first data in the first page, and mount the first page in an index tree corresponding to a disk.

After creating the first page, the electronic device may store the first data in the first page.

For example, if the electronic device created the first page as the page P30, the electronic device may store data 1 in the page P30.

In an optional embodiment, mounting the first page in the index tree corresponding to the disk may be implemented as follows: determining a first primary page corresponding to the first page in the index tree; and updating index-related information of the first page, updating a page content of the first primary page, and updating index-related information of the first primary page, so as to mount the first page in the index tree.

The index tree may be the Bw^{e}-tree described in FIG. 2. The first primary page may be an index page in the index tree.

For example, if the first page is the page P30 and the corresponding first primary page is the page P15, the electronic device may update the index-related information of the page P30, and update the page content and index-related information of the page P15, so as to mount the page P30 in the index tree.

In an embodiment of the present disclosure, the electronic device may receive a data storage request. When the data size of the first data is greater than or equal to the preset threshold, the electronic device may create a first page corresponding to the first data, store the first data in the first page, and mount the first page in the index tree corresponding to the disk. Since the electronic device may store the first data separately in the first page and then mount the first page in the index tree, there is no need to store the first data in a target page in the index tree, thereby avoiding frequent splitting operations caused on the target page by the first data being very large, and reducing the resource overhead for data storage.

Based on the embodiment shown in FIG. 3, the above data storage process is described in detail below with reference to FIG. 4.

FIG. 4 is a schematic flowchart of another data storage process provided by an exemplary embodiment of the present disclosure. Referring to FIG. 4, the method may include the following.

Step S401: Receive a data storage request.

Step S402: When a data size of a first data is greater than or equal to a preset threshold, create a first page corresponding to the first data.

It should be noted that for the specific execution process of Step S401 to Step S402, reference may be made to the specific execution process of Step S301 to Step S302, which will not be repeated here.

Step S403: Store the first data in the first page.

After creating the first page, the electronic device may store the first data in the first page.

Since the data storage request may further include the first key value and the LSN number corresponding to the first data, the electronic device may also store the first key value and the LSN number in the first page.

For example, if the electronic device created the first page as the page P30, the electronic device may store the first key value 910, data 1, and LSN1 in the page P30.

Step S404: Update index-related information of the first page.

In an optional embodiment, updating the index-related information of the first page may be implemented as follows: determining a storage address of the first page in the disk and a first key value corresponding to the first data; generating a first memory record corresponding to the first page; storing a first page identifier of the first page in association with a memory address of the first memory record in a page mapping table corresponding to the disk; and updating a state corresponding to the first page to a pending allocation state in the page mapping table.

Optionally, the electronic device may determine the storage address of the first page in the disk and store the first page to the disk.

Optionally, the electronic device may also set a pending allocation (PendingAlloc) flag in the first page in the disk. The PendingAlloc flag is used to indicate that the first page has not been mounted to the index tree.

The electronic device may generate, in the memory, a first memory record corresponding to the first page.

Since only the first data is stored in the first page, the electronic device may use the first key value corresponding to the first data as the first key value corresponding to the first page. The first memory record corresponding to the first page may include the first key value and the storage address of the first page in the disk. Optionally, the first memory record may further include the first page identifier of the first page and the memory address of the first page.

For example, if the first page is the page P30, the first page identifier of the first page may be 30. If the first key value corresponding to data 1 is 910, the electronic device may generate, in the memory, a memory record 1 corresponding to the page P30, and the memory record 1 may include the first page identifier 30, the first key value 910, the memory address of the page P30, and the storage address of the page P30 in the disk.

The memory of the electronic device may store the page mapping table corresponding to the disk. The electronic device may determine the memory address of the first memory record corresponding to the first page in the memory, and then store the first page identifier of the first page in the page mapping table and store the memory address of the first memory record in the mapping entry corresponding to the first page identifier, so that in the page mapping table, the first page identifier may point to the memory address of the first memory record.

The electronic device may store a PendingAlloc flag in the mapping entry corresponding to the first page identifier in the page mapping table, so as to update the state corresponding to the first page to the pending allocation state. The pending allocation state is used to indicate that the first page has not been mounted to the index tree.

For example, if the first page is the page P30 and the page P30 has a corresponding memory record 1, the electronic device may store the first page identifier 30 in the page mapping table and store the memory address of the memory record 1 and the PendingAlloc flag in the mapping entry corresponding to the first page identifier 30.

Step S405: Determine a first primary page corresponding to the first page in the index tree.

Since only the first data is stored in the first page, the electronic device may use the first key value corresponding to the first data as the first key value corresponding to the first page.

Optionally, the electronic device may determine a target key value range in which the first key value is located according to the first key value corresponding to the first page and according to the index tree, and then determine the index page corresponding to the target key value range as the first primary page.

For example, if the first key value corresponding to the data 1 is 910 and the data 1 is stored in the first page, i.e., the page P30, the electronic device may determine that the first key value corresponding to the page P30 is 910, and then may determine, according to the index tree, that the first key value 910 is within the target key value range [800,950). If the index page corresponding to the target key value range [800,950) is the page P15, the electronic device may determine that the first primary page corresponding to the page P30 is the page P15.

Step S406: Update the page content of the first primary page.

In an optional embodiment, updating the page content of the first primary page may be implemented as follows: generating a first index record based on the first key value corresponding to the first data and the first page identifier of the first page; and adding the first index record in the first primary page.

The first index record includes the first key value corresponding to the first data and the first page identifier.

For example, if the first page is the page P30, the first page identifier is 30, and the first key value is 910, the electronic device may generate the first index record (910, P30) according to the first page identifier 30 and the first key value 910. If the first primary page is the page P15, and the page P15 includes index records: (810, P28), (840, P27), (880, P29), the electronic device may add the first index record (910, P30) in the page P15. The page P15 may include index records: (810, P28), (840, P27), (880, P29), (910, P30).

Step S407: Update the index-related information of the first primary page.

In an optional embodiment, updating the index-related information of the first primary page may be implemented as follows: generating a first flush-to-disk record corresponding to the first primary page; and updating the memory address corresponding to the first primary page to an address of the first flush-to-disk record in the memory in the page mapping table corresponding to the disk.

The electronic device may determine the storage address of the first primary page in the disk, generate a first flush-to-disk record corresponding to the first primary page in the memory, and link the first flush-to-disk record after the target page.

The first flush-to-disk record may include the memory address of the first primary page and the storage address of the first primary page in the disk. Optionally, the first flush-to-disk record may further include at least one of the following: the memory address of the first primary page, the data-delta count, the page segment count, and the page size of the first primary page.

For example, if the first primary page is the page P15, the electronic device may determine the storage address of the page P15 in the disk and generate a flush-to-disk record 1

corresponding to the page P15, and the flush-to-disk record 1 may include the memory address of the page P15, a data-delta count of 1, a page segment count of 1, a page size of 7 KB, and the storage address of the page P15 in the disk.

Before the first flush-to-disk record is generated, the mapping entry corresponding to the first primary page identifier in the page mapping table stores the memory address of the first primary page. After generating the first flush-to-disk record corresponding to the first primary page in the memory, the electronic device may, in the page mapping table, update the memory address of the first primary page to the memory address of the first flush-to-disk record in the mapping entry corresponding to the first primary page identifier, so that the first primary page identifier in the page mapping table may point to the first flush-to-disk record.

For example, if the first primary page is the page P15 and the page P15 has a corresponding flush-to-disk record 1, the memory address of the page P15 may be updated to the memory address of the flush-to-disk record 1 in the mapping entry corresponding to the first primary page identifier 15 in the page mapping table.

Step S408: Perform flush-to-disk processing on the first primary page in the disk based on the first flush-to-disk record.

For example, if the first primary page is the page P15 and a flush-to-disk record 1 corresponding to the page P15 includes the storage address of the page P15 in the disk, the electronic device may store the page P15 in the disk according to the storage address of the page P15 in the disk, i.e., perform flush-to-disk processing on the page P15.

Step S409: Update, in the disk, a state corresponding to the first page in the first primary page to a confirmed allocation state.

The electronic device may set, in the disk, a confirmed allocation (ValidatingAlloc) flag corresponding to the first page in the first primary page. The ValidatingAlloc flag is used to indicate that the first page has been mounted to the index tree.

For example, if the first page is the page P30 and the first primary page is the page P15, after storing the page P15 to the disk, the electronic device may set a ValidatingAlloc flag corresponding to P30 in the page P15, to indicate that the page P30 has been mounted to the index tree.

Step S410: Update the state corresponding to the first page to the confirmed allocation state in the page mapping table.

Optionally, the electronic device may delete the PendingAlloc flag in the mapping entry corresponding to the first page identifier in the page mapping table, so as to update the state corresponding to the first page to the confirmed allocation state.

For example, if the first page is the page P30, the electronic device may delete the PendingAlloc flag in the mapping entry corresponding to the first page identifier 30 in the page mapping table, so as to update the state corresponding to the page P30 to the confirmed allocation state.

In the embodiment of the present disclosure, the electronic device may receive a data storage request. If the data size of the first data is greater than or equal to the preset threshold, the electronic device may create a first page corresponding to the first data, store the first data in the first page, and then update the index-related information of the first page. The electronic device may determine a first primary page corresponding to the first page in the index tree, and update the page content of the first primary page and update the index-related information of the first primary page, so as to mount the first page in the index tree. The electronic device may perform flush-to-disk processing on the first primary page in the disk according to the first flush-to-disk record, and update, in the disk, the state corresponding to the first page in the first primary page to the confirmed allocation state. The electronic device may update the state corresponding to the first page to the confirmed allocation state in the page mapping table. Since the electronic device may store the first data separately in the first page and then mount the first page to the index tree, there is no need to store the first data in a target page in the index tree, thereby avoiding frequent splitting operations caused by the first data being very large in the target page, and reducing the resource overhead for data storage.

Based on the embodiment shown in FIG. 4, the above data storage method is further described below with reference to FIG. 5.

FIG. 5 is a schematic diagram of a process of a data storage method provided by an exemplary embodiment of the present disclosure. Referring to FIG. 5, it includes steps ①② ③④⑤⑥⑦.

The electronic device may receive a data storage request 1, and the data storage request 1 may include a first key value 910, data 1, and LSN1. The electronic device may determine whether the data size of the data 1 is greater than the preset threshold.

If the data size of the data 1 is greater than the preset threshold, in step ①, the electronic device may create the page P30 in the memory and store the first key value 910, the data 1, and the LSN1 in the page P30.

In step ②, the electronic device may store the page P30 to the disk and set the PendingAlloc flag in the page P30.

The electronic device may generate a memory record 1 corresponding to the page P30. The memory record 1 may include the first page identifier 30, the first key value 910, the memory address of the page P30, and the storage address of the page P30 in the disk.

In step ③, the electronic device may store the first page identifier 30 in the page mapping table and store the memory address of the memory record 1 and the PendingAlloc flag in the mapping entry corresponding to the first page identifier 30.

In step ④, the electronic device may determine that the first primary page corresponding to the page P30 is the page P15, and update the page content of the page P15.

Specifically, the electronic device may generate the first index record (910, P30) according to the first page identifier 30 and the first key value 910, and add the first index record (910, P30) in the page P15.

In step ⑤, the electronic device may generate a flush-to-disk record 1 corresponding to the page P15 and store the page P15 to the disk according to the flush-to-disk record 1. The flush-to-disk record 1 may include the memory address of the page P15, a data-delta count of 1, a page segment count of 1, a page size of 7 KB, and the storage address of the page P15 in the disk.

After storing the page P15 to the disk, the electronic device may also set a ValidatingAlloc flag corresponding to the page P30 in the page P15, to indicate that the page P30 has been mounted to the index tree.

In step ⑥, the electronic device may update, in the page mapping table, the memory address of the page P15 to the memory address of the flush-to-disk record 1 in the mapping entry corresponding to the first primary page identifier 15.

In step ⑦, the electronic device may delete the PendingAlloc flag in the mapping entry corresponding to the first page identifier 30 in the page mapping table and delete the PendingAlloc flag in the page P30.

In an embodiment of the present disclosure, the electronic device may receive a data storage request. If the data size of the first data is greater than or equal to the preset threshold, the electronic device may create a first page corresponding to the first data, store the first data in the first page, and then update the index-related information of the first page. The electronic device may determine a first primary page corresponding to the first page in the index tree, and update the page content of the first primary page and update the index-related information of the first primary page, so as to mount the first page in the index tree. The electronic device may perform flush-to-disk processing on the first primary page in the disk according to the first flush-to-disk record, and update, in the disk, the state corresponding to the first page in the first primary page to the confirmed allocation state. The electronic device may update the state corresponding to the first page to the confirmed allocation state in the page mapping table. Since the electronic device may store the first data separately in the first page and then mount the first page to the index tree, there is no need to store the first data in a target page in the index tree, thereby avoiding frequent splitting operations caused by the first data being very large in the target page, and reducing the resource overhead for data storage.

The above FIG. 3 to FIG. 5 describe the data storage method. In the technical solutions of the present disclosure, a data deletion method is also correspondingly provided. The data deletion method is described below with reference to FIG. 6 to FIG. 8.

FIG. 6 is a schematic flowchart of a data deletion method provided by an exemplary embodiment of the present disclosure. Referring to FIG. 6, the method may include the following.

Step S601: Receive a data deletion request.

The data deletion request may be used to request deletion of second data corresponding to a second key value. The second data may be data that is greater than or equal to a preset threshold.

The data deletion request may include the second key value. Optionally, the data deletion request may further include an LSN number corresponding to the second data.

For example, the electronic device may receive a data deletion request 1, and the data deletion request 1 may include a second key value 810 corresponding to a data 2 and LSN1. The data storage request 1 is used to request deletion of the data 2.

Step S602: Determine a second page corresponding to a second data in an index tree.

Since the data deletion request may include the second key value, and the key value corresponding to the page storing the second data is the second key value, after obtaining the second key value, the electronic device may search for the page where the second key value is located in the index tree, and determine the page where the second key value is located as the second page corresponding to the second data. The second page may be a page that stores the second data separately.

For example, if the data deletion request 1 includes the second key value 810 and the key value corresponding to a page P28 is 810, the electronic device may determine, according to the second key value 810, that the second page corresponding to data 2 is the page P28.

Step S603: Delete the second page and index-related information in the index tree, so as to delete the second data.

In an optional embodiment, deleting the second page and the index-related information in the index tree may be implemented as follows: deleting the second page; determining a second primary page corresponding to the second page in the index tree; and updating the page content and the index-related information of the second primary page.

For example, if the second page is the page P28, the electronic device may delete the page P28.

When determining the second page, the electronic device finds the second page through the second index record in the second primary page corresponding to the second page, therefore, the electronic device has already determined the second primary page

corresponding to the second page when determining the second page in the index tree. For example, if the second page is the page P28, the electronic device may determine that the second primary page corresponding to the page P28 is the page P15. The electronic device may update the page content and the index-related information in the page P15, so as to delete the data 2.

In the embodiment of the present disclosure, the electronic device may receive a data deletion request, determine a second page corresponding to the second data in the index tree, and then delete the second page and the index-related information in the index tree, so as to delete the second data. Since the electronic device may delete the second page that separately stores the second data without causing frequent page merge operations, the resource overhead of deleting data is reduced.

Based on the embodiment shown in FIG. 6, the above data deletion method is described in detail below with reference to FIG. 7.

FIG. 7 is a schematic flowchart of another data deletion method provided by an exemplary embodiment of the present disclosure. Referring to FIG. 7, the method may include the following.

Step S701: Receive a data deletion request.

Step S702: Determine a second page corresponding to a second data in an index tree.

It should be noted that for the specific execution process of Step S701 to Step S702, reference may be made to the specific execution process of Step S601 to Step S602, which will not be repeated here.

Step S703: Delete the second page.

Optionally, deleting the second page may include the following two cases.

Case 1: The second page has not been flushed to the disk.

In this case, since the second page has not been flushed to the disk, both the second page and the second memory record are in the memory, and the electronic device may delete the second page and the second memory record corresponding to the second page from the memory corresponding to the disk.

The second memory record may include the second key value and the storage address of the second page in the disk. Optionally, the second memory record may further include the second page identifier of the second page and the memory address of the second page.

For example, if the second page is a page P28, a memory record 2 corresponding to the page P28 may include a second page identifier 30, a second key value 810, a memory address of the page P28, and a storage address of the page P28 in the disk.

For example, if the second page is the page P28, the page P28 has not been flushed to the disk, and the page P28 has a corresponding memory record 2, the electronic device may delete the page P28 and the memory record 2 from the memory.

Case 2: If the second page has been flushed to the disk, the second page is deleted from both the memory and the disk, respectively, and the second memory record is deleted from the memory.

In this case, since the second page has been flushed to the disk, the second page may exist simultaneously in the disk and the memory, and the second memory record is in the memory, the electronic device may delete the second page from both the memory and the disk, and delete the second memory record from the memory.

For example, if the second page is the page P28, the page P28 has been flushed to the disk, the page P28 may exist simultaneously in the disk and the memory, and the memory record 2 corresponding to the page P28 exists in the memory, the electronic device may delete the page P28 from the disk and the memory, and delete the memory record 2 from the memory.

During the process of generating the second page, the memory address of the second memory record is stored in the mapping entry corresponding to the second page identifier in the page mapping table, therefore, in the above two cases, after deleting the second page and the second memory record, the electronic device may delete the memory address of the second memory record from the mapping entry corresponding to the second page identifier in the page mapping table; however, since the second index record corresponding to the second page has not been deleted in the second primary page corresponding to the second page, the electronic device may set a PendingDealloc (PendingDealloc) flag in the mapping entry

corresponding to the second page identifier in the page mapping table corresponding to the disk, to update the state of the second page to the pending deletion state.

For example, if the second page is the page P28, the electronic device deletes a memory address of a memory record 2 from a mapping entry corresponding to a second page identifier 28 in the page mapping table, and sets a PendingDealloc flag, to update the state of the page P28 to the pending deletion state.

S704: Determine a second primary page corresponding to the second page in the index tree.

For example, if the second page is the page P28, the electronic device may determine that the second primary page corresponding to the page P28 is the page P15.

S705: Update page content of the second primary page.

In an optional embodiment, updating the page content of the second primary page may be implemented as follows: determining a second index record corresponding to the second page; and updating a state corresponding to the second index record to a confirmed deletion state in the second primary page.

The second index record may include a second key value and a second page identifier corresponding to the second page. For example, if the second page is the page P28 and the second key value is 810, the electronic device may determine that the second index record may be (810, P28).

The second primary page includes a plurality of index records, and the plurality of index records include the second index record. Optionally, the electronic device may set a confirmed deletion (ValidatingDealloc) flag on the second index record in the second primary page, to update the state corresponding to the second index record to the confirmed deletion state.

For example, if the second primary page is the page P15, the page P15 may include a plurality of index records: (810, P28), (840, P27), (880, P29), (910, P30). If the second index record is (810, P28), the electronic device sets a PendingDealloc flag on the second index record (810, P28) in the page P15, to update the state corresponding to the second index record (810, P28) to the confirmed deletion state. After the update, the page P15 may include

a plurality of index records: (810, P28, PendingDealloc flag), (840, P27), (880, P29), (910, P30).

S706: Update index-related information of the second primary page.

In an optional embodiment, updating the index-related information of the second primary page may be implemented as follows: generating a second flush-to-disk record corresponding to the second primary page; and updating a memory address of the second primary page to a memory address of the second flush-to-disk record in a page mapping table corresponding to the disk.

The electronic device may determine a storage address of the second primary page in the disk, flush the second primary page to the disk, and generate the second flush-to-disk record. The second flush-to-disk record may include the storage address of the second primary page in the disk. Optionally, the second flush-to-disk record may further include at least one of the following: the memory address of the second primary page, a data-delta count, a page segment count, and a page size of the second primary page.

For example, if the second primary page is the page P15, the electronic device may generate flush-to-disk record 2 corresponding to the page P15, and flush-to-disk record 2 may include the memory address of the page P15, a data-delta count of 0, a page segment count of 1, a page size of 7 KB, and the storage address of the page P15 in the disk.

Before the second flush-to-disk record is generated, the memory address of the second primary page is stored in the mapping entry corresponding to the second primary page identifier in the page mapping table, therefore, after generating the second flush-to-disk record corresponding to the second primary page in the memory, the electronic device may update, in the page mapping table, the memory address of the second primary page to the memory address of the second flush-to-disk record in the mapping entry corresponding to the second primary page identifier, so that the second primary page identifier in the page mapping table may point to the second flush-to-disk record.

For example, if the second primary page is the page P15 and the page P15 has a corresponding flush-to-disk record 2, in the page mapping table, the memory address of the

page P15 is updated to the memory address of flush-to-disk record 2 in the mapping entry corresponding to the second primary page identifier 15.

S707: Update a state corresponding to the second page to the confirmed deletion state in the page mapping table.

In step S703, the electronic device sets the PendingDealloc flag in the mapping entry corresponding to the second page identifier in the page mapping table, therefore, the electronic device may delete the PendingDealloc flag, to update the state corresponding to the second page to the confirmed deletion state.

Optionally, the electronic device may further delete the second page identifier, or recycle the second page identifier as a free page identifier.

For example, if the second page is the page P28 and the page P28 has a corresponding memory record 2, the electronic device may delete the PendingDealloc flag in the page mapping table, to update the state corresponding to the page P28 to the confirmed deletion state.

In the embodiment of the present disclosure, the electronic device may receive a data deletion request, determine a second page corresponding to second data in the index tree, and delete the second page. The electronic device may determine a second primary page corresponding to the second page in the index tree, update the page content and the index-related information of the second primary page, and delete mapping information corresponding to the second page in the page mapping table. Since the electronic device may delete the second page that separately stores the second data without causing frequent page merge operations, the resource overhead of deleting data is reduced.

Based on the embodiment shown in FIG. 6 or FIG. 7, the above data deletion process is further described below with reference to FIG. 8.

FIG. 8 is a schematic diagram of a process of a data deletion method according to an exemplary embodiment of the present disclosure. Referring to FIG. 8, the process includes steps ①②③④⑤⑥⑦.

The electronic device may receive data deletion request 1, and data deletion request 1 may include a second key value 810 and LSN1. The data deletion request 1 is used for data 2 corresponding to the second key value 810 and the LSN1.

In step ①, the electronic device may determine, based on the second key value 810, that the second page corresponding to the data 2 is the page P28.

In step ②, if the page P28 exists in both the memory and the disk, the electronic device may delete the page P28 from the disk and the memory, and delete a memory record 2 from the memory. The memory record 2 may include the second page identifier 28, the second key value 810, the memory address of the page P28, and the storage address of the page P28 in the disk.

In step ③, the electronic device may delete the memory address of the memory record 2 from the mapping entry corresponding to the second page identifier 28 in the page mapping table, and set a PendingDealloc flag, to update the state of the page P28 to the pending deletion state.

In step ④, the electronic device may determine that the second primary page corresponding to the page P28 is the page P15, and update the page content of the page P15.

Specifically, the electronic device may set a PendingDealloc flag on the second index record (810, P28) in the page P15, to update the state corresponding to the second index record (810, P28) to the confirmed deletion state.

In step ⑤, the electronic device may generate a flush-to-disk record 2 corresponding to the page P15, and perform flush-to-disk processing on the page P15. The flush-to-disk record 2 may include the memory address of the page P15, a data-delta count of 0, a page segment count of 1, a page size of 7 KB, and the storage address of the page P15 in the disk.

In step ⑥, the electronic device may update the memory address of the page P15 to the memory address of the flush-to-disk record 2 in the mapping entry corresponding to the second primary page identifier 15 in the page mapping table.

In step ⑦, the electronic device may delete the PendingDealloc flag in the page mapping table, to update the state corresponding to the page P28 to the confirmed deletion state, that is, to release the second page identifier 28.

In the embodiment of the present disclosure, the electronic device may receive a data deletion request, determine a second page corresponding to second data in the index tree, and delete the second page. The electronic device may determine a second primary page corresponding to the second page in the index tree, update the page content and the index-related information of the second primary page, and delete mapping information corresponding to the second page in the page mapping table. Since the electronic device may delete the second page that separately stores the second data without causing frequent page merge operations, the resource overhead of deleting data is reduced.

When the electronic device performs merge processing on pages in the index tree, concurrent updates to pages may occur. In the technical solutions of the present disclosure, an update linking mechanism is further provided, so that updates to pages are not affected while merge processing is being performed on the pages. The update linking mechanism is described below with reference to FIG. 9.

FIG. 9 is a schematic flowchart of an update linking mechanism according to an exemplary embodiment of the present disclosure. Referring to FIG. 9, the mechanism includes the following.

S901: Receive a data merge request.

The data merge request is used to request merging a third page and at least one update record of the third page.

The update record may be the data-delta record described in the embodiment of FIG. 2. Any update record may include: update data for the third page, a memory address of a preceding update record or the third page, a data-delta count, a page segment count, and a page size.

Optionally, when a first record count of the at least one update record of the third page is greater than or equal to a first count threshold, the electronic device may generate the data merge request; or the electronic device may receive the data merge request in response to a user operation.

The first count threshold is preset manually. For example, the first count threshold is 2.

For example, when the first record count of the at least one update record of the page P2[400,700) is 2, equal to the first count threshold of 2, the electronic device may generate a data merge request 1; or the electronic device may receive a data merge request 1 in response to a user operation. The data merge request 1 may be used to request merging the page P2[400,700) and corresponding update record 1 and update record 2.

S902: Generate a merge record corresponding to the third page based on the data merge request, and mount the merge record on the third page.

The merge record may be used to mount a next update record of the third page. The next update record refers to an update record newly generated after the at least one existing update record.

The electronic device may generate, according to the data merge request, the merge record corresponding to the third page in the memory.

For example, if the third page is the page P2[400,700), the electronic device may generate, according to data merge request 1, a merge record 1 corresponding to the page P2[400,700) in the memory.

In an optional embodiment, mounting the merge record on the third page may be implemented as follows: determining a second update record among the at least one update record; adding a memory address of the second update record in the merge record; and updating a mapping content corresponding to a third page identifier of the third page to a memory address of the merge record in a page mapping table corresponding to the disk.

The second update record is the last update record among the at least one update record.

For example, if the third page is the page P2[400,700), which has corresponding update record 1 and update record 2, and if update record 2 is the last update record, the electronic device may determine, from the update record 1 and the update record 2, that the second update record is the update record 2. The electronic device may add the memory address 2 of the update record 2 to the merge record 1.

Optionally, the electronic device may further determine the first record count of the at least one update record, and add the first record count to the merge record. The first record

count is used to represent the number of update records corresponding to the third page at that time.

For example, if the third page is the page P2[400,700), and the page P2[400,700) has corresponding update record 1 and update record 2, the first record count is 2, and the electronic device may add the first record count of 2 to the merge record 1.

The electronic device may determine the third page identifier of the third page, and in the mapping entry corresponding to the third page identifier in the page mapping table, update the mapping content corresponding to the third page identifier to the memory address of the merge record in the memory.

For example, if the third page is the page P2[400,700), the electronic device may determine that the third page identifier is 2. If the mapping entry corresponding to the third page identifier (i.e., the page identifier 2) in the page mapping table stores the memory address 2 of the update record 2, the electronic device may update, in the page mapping table, the mapping content corresponding to the third page identifier, i.e., the memory address 2 of the update record 2, to the memory address of the merge record 1 in the mapping entry corresponding to the third page identifier (i.e., the page identifier 2).

S903: Perform merge processing on the third page and the at least one update record to obtain a merged page.

Optionally, the electronic device may modify the page content of the third page according to the at least one update record, to perform merge processing on the third page and the at least one update record to obtain the merged page.

For example, if the third page is the page P2[400,700), the update record 1 is adding data corresponding to a key value 515 to the page P2[400,700), and the update record 2 is adding data corresponding to a key value 455 to the page P2[400,700), the electronic device may add the data corresponding to the key value 515 to the page P2[400,700) according to the update record 1, and add the data corresponding to the key value 455 to the page P2[400,700) according to the update record 2, to obtain the updated page P2[400,700), which is the merged page.

S904: Determine the memory address of the merged page.

For example, if the merged page is the updated page P2[400,700), the electronic device may determine the memory address of the updated page P2[400,700).

S905: Update the memory address of the second update record to the memory address of the merged page in the merge record.

When generating the merge record (i.e., before performing the page merge), the electronic device added the memory address of the second update record to the merge record, therefore, after performing merge processing on the third page and the at least one update record to obtain the merged page, the electronic device may update the memory address of the second update record in the merge record to the memory address of the merged page, so that the merge record is mounted on the merged page.

For example, if the merge record 1 includes the memory address 2 of the update record 2, and the merged page is the updated page P2[400,700), the electronic device may update the memory address 2 of the update record 2 in the merge record 1 to the memory address of the updated page P2[400,700).

S906: Update the first record count to 0 in the merge record.

After performing merge processing on the third page and the at least one update record to obtain the merged page, the merged page does not have any corresponding update record at this time, the electronic device may update the first record count to 0 in the merge record, to indicate that no update record exists between the merge record and the merged page.

For example, if the first record count in the merge record 1 is 2, after obtaining the merged page, the first record count in the merge record 1 is updated from 2 to 0.

S907: Receive an update request corresponding to the third page.

It should be noted that the timing of executing steps S907 to S909 includes two cases.

Case 1: While executing the above steps S903 to S905, the electronic device concurrently executes steps S907 to S909.

Case 2: After completing steps S903 to S906, the electronic device sequentially executes steps S907 to S909.

Whether in Case 1 or Case 2, the electronic device may receive an update request corresponding to the third page.

For example, if the third page is the page P2[400,700), the electronic device may receive an update request 3 corresponding to the page P2[400,700), and the update request 3 may be a request to add data corresponding to a key value 650 to the page P2[400,700).

S908: Generate a target update record corresponding to the third page.

After receiving the update request corresponding to the third page, the electronic device generates, in the memory, the target update record corresponding to the third page according to the update request.

The target update record may be the data-delta record described in the embodiment of FIG. 2. The target update record may include update data for the third page. Optionally, the target update record may further include at least one of the following: a data-delta count, a page segment count, and a page size.

It should be noted that the data-delta count in the target update record is equal to the first record count in the merge record plus one, as follows.

(1) If, when generating the target update record corresponding to the third page, the first record count in the merge record has not yet been updated to 0 (i.e., Case 1), the data-delta count in the target update record is the first record count + 1.

For example, if the first record count in the merge record 1 is 2, the data-delta count in the target update record is 3.

(2) If, when generating the target update record corresponding to the third page, the first record count in the merge record has already been updated to 0 (i.e., Case 2), the data-delta count in the target update record is 1, that is, the first record count (0) + 1.

For example, if the third page is the page P2[400,700), the electronic device may receive an update request 3 corresponding to the page P2[400,700), which is a request to add data corresponding to a key value 650 to the page P2[400,700), the electronic device may generate an update record 3, i.e., the target update record, according to the update request 3. The update record 3 may include the update data for the third page, i.e., the data corresponding to the key value 650, a data-delta count of 3, a page segment count of 1, and a page size of 8 KB.

S909: Add the memory address of the merge record to the target update record, and update the memory address corresponding to the merge record to the memory address of the target update record in the page mapping table corresponding to the disk.

After generating the target update record corresponding to the third page, the electronic device may add the memory address of the merge record to the target update record, so that the target update record is linked after the merge record.

For example, if the electronic device generates the target update record as an update record 3, the memory address of the merge record 1 is added to the update record 3, so that the update record 3 is linked after the merge record 1.

In step S902, the electronic device has already updated, in the page mapping table corresponding to the disk, the mapping content corresponding to the third page identifier of the third page to the memory address of the merge record in the memory, therefore, after generating the target update record, the electronic device may update the memory address corresponding to the merge record to the memory address of the target update record in the page mapping table corresponding to the disk.

For example, if the third page is the page P2[400,700) and the target update record is the update record 3, the electronic device may update the memory address of the merge record 1 to the memory address of the update record 3 in the mapping entry corresponding to the third page identifier (i.e., the page identifier 2) in the page mapping table.

It should be noted that, when in Case 1, steps S903 to S905 and steps S907 to S909 are executed concurrently, and then step S906 is executed sequentially, there is a situation where step S908 is executed before step S906, that is, there is a situation where the first record count in the merge record 1 is 2 and the data-delta count in the target update record is 3. Therefore, after executing step S906 to update the first record count in the merge record 1 to 0, the data-delta count in the target update record is updated to 1, that is, the first record count (0) + 1.

In the embodiment of the present disclosure, the electronic device may receive a data merge request. The electronic device may generate a merge record corresponding to the third page based on the data merge request, and mount the merge record on the third page. The electronic device may perform merge processing on the third page and the at least one update

record to obtain a merged page, and determine the memory address of the merged page, and then update the memory address of the second update record in the merge record to the memory address of the merged page. The electronic device may update the first record count to 0 in the merge record. The electronic device may receive an update request corresponding to the third page, generate a target update record corresponding to the third page, then add the memory address of the merge record to the target update record, and update the memory address corresponding to the merge record to the memory address of the target update record in the page mapping table corresponding to the disk. Since the merge record may be generated and mounted on the third page before performing merge processing on the third page and the at least one update record, when performing merge processing on the third page and the at least one update record while concurrently updating the third page, the newly generated target update record is linked after the merge record, so that the update of the third page is not delayed while performing merge processing on the third page and the at least one update record, thereby improving the efficiency of processing pages.

Based on the embodiment shown in FIG. 9, the above update linking mechanism is further described below through specific examples with reference to FIG. 10.

FIG. 10 is a schematic diagram of a process of an update linking mechanism according to an exemplary embodiment of the present disclosure. Referring to FIG. 10, the process includes steps ①②③④⑤⑥⑦.

The electronic device may receive a data merge request 1. The data merge request 1 is used to request merging the page P2[400,700) and corresponding update record 1 and update record 2.

In step ①, the electronic device may generate, according to the data merge request 1, a merge record 1 corresponding to the page P2[400,700) in the memory, and add the memory address 2 of the update record 2 and the first record count of 2 to the merge record 1.

In step ②, the electronic device may update the mapping content corresponding to the third page identifier to the memory address of the merge record 1 in the mapping entry corresponding to the third page identifier (i.e., the page identifier 2) in the page mapping table.

In step ③, the electronic device may perform merge processing on the page P2[400,700), the update record 1, and the update record 2 to obtain the merged page. Specifically, the electronic device may add the data corresponding to the key value 515 to the page P2[400,700) according to the update record 1, and add the data corresponding to the key value 455 to the page P2[400,700) according to the update record 2, to obtain the merged page, i.e., the updated page P2[400,700).

In step ④, the electronic device may determine the memory address of the merged page (i.e., the updated page P2[400,700)), and update the memory address 2 of the update record 2 in the merge record 1 to the memory address of the merged page.

In step ⑤, the electronic device may update the first record count from 2 to 0 in the merge record 1.

If, when executing steps ③, ④, and ⑤, there is an update request corresponding to the page P2[400,700), the electronic device may execute steps ⑥ and ⑦ in parallel.

In step ⑥, the electronic device may receive an update request 3 corresponding to the page P2[400,700), generate, according to the update request 3, a target update record for the page P2[400,700), i.e., an update record 3, and add the memory address of the merge record 1 to the update record 3. The update record 3 may include: the update data for the third page, i.e., the data corresponding to the key value 650, the memory address of the merge record 1, a data-delta count of 3, a page segment count of 1, and a page size of 8 KB.

In step ⑦, the electronic device may update the memory address of the merge record 1 to the memory address of the update record 3 in the mapping entry corresponding to the third page identifier (i.e., the page identifier 2) in the page mapping table.

In the embodiment of the present disclosure, the electronic device may receive a data merge request. The electronic device may generate a merge record corresponding to the third page based on the data merge request, and mount the merge record on the third page. The electronic device may perform merge processing on the third page and the at least one update record to obtain a merged page, and determine the memory address of the merged page, and then update the memory address of the second update record in the merge record to the memory address of the merged page. The electronic device may update the first record count

to 0 in the merge record. The electronic device may receive an update request corresponding to the third page, generate a target update record corresponding to the third page, then add the memory address of the merge record to the target update record, and update the memory address corresponding to the merge record to the memory address of the target update record in the page mapping table corresponding to the disk. Since the merge record may be generated and mounted on the third page before performing merge processing on the third page and the at least one update record, when performing merge processing on the third page and the at least one update record while concurrently updating the third page, the newly generated target update record is linked after the merge record, so that the update of the third page is not delayed while performing merge processing on the third page and the at least one update record, thereby improving the efficiency of processing pages.

In the technical solution of the present disclosure, a garbage collection method is further provided. The garbage collection method is described below with reference to FIG. 11.

FIG. 11 is a schematic flowchart of a garbage collection method according to an exemplary embodiment of the present disclosure. Referring to FIG. 11, the method may include the following.

S1101: Determine, in the disk, at least one target data block to be subjected to garbage collection.

Optionally, the electronic device may maintain a current garbage collection point (Garbage Collection Point, GCP). The GCP is a logical block number of the next data block to be collected that was recorded during the last garbage collection, that is, the logical block number of the first data block in the current garbage collection. The first data block refers to a data block to be firstly subjected to collection processing in the current garbage collection.

It should be noted that, in any garbage collection process, the GCP dynamically changes as the garbage collection progresses.

Optionally, when the size of the free storage space in the disk is less than or equal to a first threshold, or when the garbage ratio in the disk is greater than or equal to a second threshold, the electronic device may execute the steps shown in FIG. 11.

Both the first threshold and the second threshold may be preset manually.

In an optional embodiment, the electronic device determining the garbage ratio in the disk may be implemented as follows: determining a valid data size V in the disk; determining a garbage size G in the disk; and determining a ratio of the garbage size G to the valid data size V as the garbage ratio K. Where, V and G are both integers greater than or equal to 0, and K is a decimal between 0 and 1.

Optionally, the electronic device may determine the valid data size V by the following formula (1):

$$V = T * B - I \qquad \text{Formula (1)}$$

Where, T represents a logical block number of the latest data block that has been flushed to the disk; B represents the size of one data block; and I represents the total size of historical invalid data.

The total size of the historical invalid data refers to the total size of all invalid data among the plurality of data blocks from the first data block stored in the disk to the latest data block that has been flushed to the disk.

Formula (1) indicates: the logical block number of the latest data block that has been flushed to the disk multiplied by the size of one data block equals the total size of all data blocks that have been stored in the disk, and then subtracting the total size of the historical invalid data yields the size of valid data currently present in the disk.

For example, if the logical block number T of the latest data block that has been flushed to the disk is 1000, the size B of one data block is 2 MB (MB), the disk currently stores 750 data blocks corresponding to logical block numbers 251 to 1000, and the total size I of the historical invalid data (i.e., the total size of all invalid data among the 1000 data blocks corresponding to logical block numbers 1 to 1000) is 1500 MB, the valid data size among the 750 data blocks currently in the disk may be determined to be 1000 * 2 - 1500 = 500 MB.

Optionally, the electronic device may determine a garbage size G by the following formula (2):

$$G = I - (H - 1) * B \qquad \text{Formula (2)}$$

Where, I represents the total size of the historical invalid data; H represents the logical block number of the first data block; and B represents the size of one data block.

In formula (2), (H - 1) * B is the size of the plurality of invalid data blocks before the first data block from which garbage collection currently starts.

Formula (2) indicates: the total size of the historical invalid data minus the total size of the plurality of invalid data blocks before the first data block equals the total garbage size in the disk from the first data block to the latest data block that has been flushed to the disk.

The plurality of data blocks before the first data block are all invalid data blocks (regardless of whether the plurality of data blocks are currently in the disk).

For example, if the logical block number T of the latest data block that has been flushed to the disk is 1000, the size B of one data block is 2 MB (MB), the total size I of the historical invalid data (i.e., the total size of all invalid data among the 1000 data blocks corresponding to logical block numbers 1 to 1000) is 1500 MB, the disk currently stores 750 data blocks corresponding to logical block numbers 251 to 1000, and the logical block number of the first data block is 280, the garbage size G among the 720 data blocks with logical block numbers 280 to 1000 currently in the disk may be determined to be 1500 - (280 - 1) * 2 = 942 MB.

Optionally, the electronic device may determine the garbage ratio K by the following formula (3):

$$K = G / (V + G) \qquad \text{Formula (3)}$$

Where, G is the garbage size in the disk; and V is the valid data size in the disk.

For example, if G is 942 MB and V is 500 MB, the garbage ratio K = 942 / (942 + 500) = 65%.

For any data block, the data block may include a plurality of page segments.

For any data block, the data block has a corresponding logical block number. For example, if a logical block number of a data block is 005, the data block is represented as a data block 005.

In an optional embodiment, determining, in the disk, the at least one target data block to be subjected to garbage collection may be implemented as follows: determining a first data block in the disk; determining a second data block on which a data operation is being performed in the disk; and determining a data block between the first data block and the second data block in the disk as the at least one target data block.

It should be noted that the at least one target data block includes the first data block but does not include the second data block.

Optionally, the electronic device may determine a garbage collection point, and determine the first data block in the disk based on the garbage collection point. For example, if the garbage collection point is 280, a data block 280 corresponding to a logical block number 280 is determined in the disk as the first data block.

For example, the electronic device may determine, in the disk, that the second data block on which a data operation is currently being performed is a data block 1001. If the first data block is a data block 280, the electronic device may determine, in the disk, 720 data blocks between the data block 280 and the data block 1001 (i.e., the data block 280, a data block 281, ..., a data block 1000) as 720 target data blocks.

S1102: Determine a valid page segment in the at least one target data block, and store the valid page segment in a buffer.

In an optional embodiment, determining the valid page segment in the at least one target data block may be implemented as follows: determining a segment state of each page segment in the at least one target data block; and determining a page segment in the valid state among the at least one target data block as a valid page segment.

For any page segment, the segment state of the page segment may be either a valid state or an invalid state.

In the disk, any page corresponds to a plurality of page segments, among the plurality of page segments, a later-stored page segment includes the storage address of an earlier-stored page segment, and the plurality of page segments may form a page segment chain corresponding to the page, therefore, in an optional embodiment, for any page segment, the electronic device may determine whether the page segment is in the corresponding page segment chain. If yes, the segment state of the page segment is the valid state; if no, the segment state of the page segment is the invalid state.

For example, assuming the page P2 has corresponding P2 page segment 0, P2 page segment 1, and P2 page segment 2, the P2 page segment 1 includes the storage address of the P2 page segment 0, and the P2 page segment 2 includes the storage address of the P2 page

segment 1, therefore, a page segment chain 1 corresponding to the page P2 includes the P2 page segment 0, the P2 page segment 1, and the P2 page segment 2. For the P2 page segment 0, the electronic device may determine whether the P2 page segment 0 is in page segment chain 1. If yes, the segment state of the P2 page segment 0 is the valid state; if no, the segment state of the P2 page segment 0 is the invalid state.

Optionally, the segment state of a page segment is determined based on the page mapping table, which includes the following three cases.

Case 1: If the mapping entry corresponding to the page identifier is empty, all page segments corresponding to the page identifier are invalid page segments.

Case 2: For a page that separately stores large data (the page may also be regarded as one page segment), if the storage address in the mapping entry corresponding to the page identifier is different from the storage address of the page, the page is an invalid page segment; if they are the same, the page is a valid page segment.

Case 3: For any page segment, the segment state of the page segment is determined based on the storage address of the page segment and a first storage address of a corresponding fourth page segment.

For any page segment, determining the segment state of the page segment may be implemented as follows: determining a fourth page corresponding to the page segment, where the page segment is obtained after a flush-to-disk is performed on the fourth page; determining the latest flush-to-disk record corresponding to the fourth page, where the first storage address is the earliest flush address of the fourth page in the disk; when the storage address of the page segment is greater than or equal to the first storage address, determining the segment state as the valid state; and when the storage address of the page segment is less than the first storage address, determining the segment state as the invalid state.

In the disk, any page corresponds to a plurality of page segments, therefore, the electronic device may determine the fourth page corresponding to the page segment.

Optionally, any flush-to-disk record, data-delta record, or stub record corresponding to the fourth page includes the first storage address corresponding to the fourth page, therefore, the earliest flush-to-disk record may include the first storage address corresponding to the

fourth page. The first storage address is the earliest flush address of the current valid segment of the fourth page in the disk.

The electronic device may determine the latest flush-to-disk record corresponding to the fourth page, and determine the first storage address in the latest flush-to-disk record. If the storage address of the page segment is greater than or equal to the first storage address, it indicates that the page segment was stored after the valid page segment corresponding to the current earliest valid first storage address, therefore, the segment state of the page segment may be determined to be the valid state; and if the storage address of the page segment is less than the first storage address, it indicates that the page segment was stored before the valid page segment corresponding to the current earliest valid first storage address, therefore, the segment state is determined to be the invalid state.

For example, the electronic device may determine that the fourth page corresponding to the page segment 0 is the page P2. Assuming the page P2 has corresponding P2 page segment 0, P2 page segment 1, and P2 page segment 2, if the latest flush-to-disk record corresponding to the page P2 is a flush-to-disk record 2, the electronic device may determine, in the flush-to-disk record 2, that the first storage address corresponding to the page P2 is a storage address 1. Assuming the storage address 1 is the storage address of the P2 page segment 1, then, for the P2 page segment 2, if the storage address of the P2 page segment 2 is greater than or equal to the storage address 1, the segment state of the P2 page segment 2 may be determined to be the valid state, and the P2 page segment 2 may be determined to be a valid page segment; and if the storage address of the page segment 0 is less than the storage address 1, the segment state of the page segment 0 may be determined to be the invalid state.

For any target data block, the electronic device may determine a valid page segment in the target data block, and store the valid page segment in the buffer.

For example, if there are 720 target data blocks between the data block 280 and the data block 1001, the electronic device may determine 2 valid page segments in the data block 280 and copy the 2 valid page segments to the buffer; determine 0 valid page segments in a data block 281; ...; determine 1 valid page segment in data block 1000 and copy the 1 valid page segment to the buffer.

S1103: Delete the at least one target data block from the disk.

After the buffer is filled, the plurality of page segments in the buffer may be repackaged in data blocks and stored to the disk. The data blocks may record the total size of historical invalid data and corresponding logical block numbers. The data blocks may also record a data start point (Data Start Point, DSP), that is, the logical block number of the next data block to be collected.

After the valid page segments in the at least one target data block have all been re-stored to the disk, the at least one target data block may be deleted, that is, when the data start point recorded in the latest data block that has been written to the disk is the logical block number of the last target data block, the at least one target data block may be deleted.

For example, if the latest data block that has been written to the disk is a data block 1300, and the data start point recorded in the data block 1300 is 1001, it indicates that the valid page segments in the 720 target data blocks between the data block 280 and the data block 1001 in the current garbage collection have all been re-stored to the disk, and the electronic device may delete 720 target data between the data block 280 and the data block 1001 from the disk.

It should be noted that a data start point recorded in a data block is the current garbage collection point. For example, during the current garbage collection process, the data start point recorded in the data block 1300 is 1001, then in the next garbage collection, the electronic device may determine, based on the data start point 1001, that the current garbage collection starting point is 1001, and start garbage collection processing from the data block 1001.

In an embodiment of the present disclosure, the electronic device may determine, in the disk, at least one target data block to be subjected to garbage collection, determine a valid page segment in the at least one target data block, store the valid page segment in a buffer, and delete the at least one target data block from the disk. Through the technical solution of the present disclosure, when performing garbage collection, it is not necessary to read metadata information in all data blocks to determine invalid data to be deleted and valid data to be retained, thereby improving the efficiency of garbage collection.

During operation of the electronic device, a fault may occur in the electronic device, resulting in an inability to read and write data based on the index tree, therefore, recovery processing is required. In the technical solution of the present disclosure, a data recovery method is further provided. The data recovery method is described below with reference to FIG. 12.

FIG. 12 is a schematic flowchart of a data recovery method according to an exemplary embodiment of the present disclosure. Referring to FIG. 12, the method may include the following.

S1201: Determine a data block count in a disk.

For example, the electronic device may determine that the data block count in the disk is 2000.

S1202: When the data block count is greater than or equal to a preset count, store a page mapping table corresponding to the disk.

Optionally, the preset count may be artificially preset. For example, the preset count may be 1900.

When the data block count is greater than or equal to the preset count, the electronic device may store, in a dedicated buffer, the page mapping table corresponding to the disk. For example, if the data block count is 2000 and the preset count is 1900, the electronic device may determine, in the memory, a page mapping table currently in use, and store the page mapping table 1 in the dedicated buffer.

It should be noted that when the page mapping table is being stored in the dedicated buffer, the current page mapping table in the memory does not cease to be used and may be continuously updated. Therefore, when storage of the page mapping table in the dedicated buffer is completed, the stored page mapping table may differ from part of the contents in the current page mapping table in the memory, that is, the page mapping table stored in the dedicated buffer is a historical version of the current page mapping table in the memory.

When the page mapping table is stored in the dedicated buffer based on the current page mapping table, a mapping content included in mapping entries in the current page mapping table is preserved together as is. The mapping content may include a PendingAlloc flag, a

PendingDealloc flag, a ValidatingAlloc flag, a ValidatingDealloc flag, storage addresses, and the like.

S1203: Determine a maximum page identifier in the page mapping table, a third data block that has most recently completed flushing when storage of the page mapping table begins, and a fourth data block on which a data operation is being performed when storage of the page mapping table is completed.

For example, the electronic device may scan the page mapping table and determine that the maximum page identifier in the page mapping table is 30; may determine, in the disk, that the third data block that has most recently completed flushing when storage of the page mapping table begins is a data block 50; and may determine, in the disk, that the fourth data block on which a data operation is being performed when storage of the page mapping table is completed is a data block 60.

S1204: Generate checkpoint information based on the maximum page identifier, the third data block, and the fourth data block, and store the checkpoint information.

The checkpoint information may include the maximum page identifier, an identifier of the third data block, and an identifier of the fourth data block. The identifier of the third data block and the identifier of the fourth data block both refer to logical block numbers.

Optionally, in the checkpoint information, the identifier of the third data block may be determined as a recovery start point, indicating that when fault recovery processing is required, recovery starts from the recovery start point; and the identifier of the fourth data block may be used as a maximum checkpoint block number, that is, a recovery end point during fault recovery processing.

Optionally, the electronic device may store the checkpoint information in the page mapping table that has already been stored in the dedicated buffer.

For example, the electronic device may generate checkpoint information 1 based on the maximum page identifier 30, the third data block being the data block 50, and the fourth data block being the data block 60. The checkpoint information 1 may include the maximum page identifier 30, the identifier 50 of the third data block, and the identifier 60 of the fourth data

block. The electronic device may store the checkpoint information 1 in the page mapping table 1 that has already been stored in the dedicated buffer.

Optionally, the electronic device may save the page mapping table and corresponding checkpoint information at different historical moments, therefore, there may be multiple versions of the page mapping tables and the corresponding checkpoint information. Generally, the latest version and the second-latest version of the page mapping table and the corresponding checkpoint information may be retained.

S1205: When a disk fault is determined, obtain the page mapping table and the checkpoint information.

Optionally, since the electronic device may store the latest version and the second-latest version of the page mapping table and checkpoint information, the electronic device may first obtain the latest version of the page mapping table and checkpoint information, and perform fault recovery processing on the disk according to the latest version of the page mapping table and checkpoint information. If the recovery fails, the electronic device may obtain the second-latest version of the page mapping table and checkpoint information, and perform fault recovery processing on the disk according to the second-latest version of the page mapping table and checkpoint information.

For example, the electronic device may obtain the latest version of the page mapping table 1 and checkpoint information 1. The checkpoint information 1 may include the maximum page identifier 30, the identifier 50 of the third data block, and the identifier 60 of the fourth data block.

S1206: Perform fault recovery processing on the disk based on the page mapping table and the checkpoint information.

Optionally, performing fault recovery processing on the disk may be implemented as follows: determining, in the checkpoint information, the identifier of the third data block and the identifier of the fourth data block; determining, in the disk, the third data block based on the identifier of the third data block, and determining the fourth data block based on the identifier of the fourth data block; and loading at least one data block between the third data

block and the fourth data block in a buffer, and updating the current page mapping table in the memory based on a plurality of page segments in the at least one data block.

Optionally, there may be at least two buffers. The at least two buffers may be used alternately.

For example, if the electronic device determines, in the checkpoint information 1, that the identifier of the third data block is 50 and the identifier of the fourth data block is 60, the electronic device may determine, based on the identifier 50 of the third data block, that the third data block is the data block 50, and may determine, based on the identifier 60 of the fourth data block, that the fourth data block is the data block 60. The electronic device may load 10 data blocks between the data block 50 and the data block 60 in the buffer, and update the current page mapping table in the memory based on a plurality of page segments in the 10 data blocks.

Optionally, for any page segment, the page segment has a corresponding page identifier, and the electronic device may scan the page segment and update the page mapping table based on the page segment, which may include the following three cases.

Case 1: The page segment includes a PendingAlloc flag or a PendingDealloc flag.

In this case, since the page segment includes the PendingAlloc flag or the PendingDealloc flag, it indicates that the page segment has not yet been allocated or deleted, and the PendingAlloc flag or the PendingDealloc flag may be set in the mapping entry corresponding to the page identifier in the current page mapping table.

Case 2: The page segment includes a ValidatingAlloc flag, and the PendingAlloc flag is present in the mapping entry corresponding to the page identifier in the current page mapping table.

In this case, since the page segment already includes the ValidatingAlloc flag, it indicates that allocation of the page segment has been completed, and the PendingAlloc flag may be deleted from the mapping entry corresponding to the page identifier in the current page mapping table.

Case 3: The page segment includes a ValidatingDealloc flag, and the PendingDealloc flag is present in the mapping entry corresponding to the page identifier in the current page mapping table.

In this case, since the page segment already includes the ValidatingDealloc flag, it indicates that the ValidatingDealloc flag has already been marked for the page segment, and the PendingDealloc flag may be deleted from the mapping entry corresponding to the page identifier in the current page mapping table.

In addition, it should be supplemented that since each page in the index tree follows the page lifecycle shown in FIG. 13, and FIG. 13 is a schematic diagram of a page lifecycle according to an exemplary embodiment of the present disclosure. Referring to FIG. 13, a page segment containing a PendingAlloc flag or a PendingDealloc flag is typically located before a page segment containing the corresponding ValidatingAlloc flag or ValidatingDealloc flag. However, during garbage collection, the page segment including the PendingAlloc flag or the PendingDealloc flag may be re-stored after the page segment including the corresponding ValidatingAlloc flag or the ValidatingDealloc flag. Therefore, the following Case 4 and Case 5 are supplemented for updating the page mapping table:

Case 4: The page segment includes a PendingAlloc flag, but the mapping content in the mapping entry corresponding to the page identifier in the current page mapping table is non-empty.

In this case, the mapping content in the mapping entry corresponding to the page identifier in the current page mapping table may be updated to the storage address of the page segment, and the PendingAlloc flag is ignored.

Case 5: The page segment includes a PendingDealloc flag, but the mapping entry corresponding to the page identifier in the current page mapping table is empty.

In this case, no additional processing is required. Since when the page segment includes the PendingDealloc flag, the mapping entry corresponding to the page identifier in the current page mapping table is intended to be set to empty.

Optionally, after scanning of the plurality of page segments in the at least one data block is completed, the current mapping table in the memory may be re-checked and updated, which may include the following three cases.

Case 1: The mapping entry corresponding to the page identifier in the current mapping table is empty.

In this case, it indicates that this page identifier is a free page identifier, and the page identifier may be collected.

Case 2: The mapping entry corresponding to the page identifier in the current mapping table includes a PendingAlloc flag.

In this case, the PendingDealloc flag is set in the mapping entry corresponding to the page identifier. The reason is: during fault recovery, data at a historical moment is recovered, and at the historical moment, the state corresponding to the page is the pending allocation state; whereas at the current moment, the page may no longer need to be allocated, and therefore the PendingDealloc flag is set in the mapping entry corresponding to the page identifier.

Case 3: The mapping entry corresponding to the page identifier in the current mapping table includes a PendingDealloc flag.

In this case, it indicates that the page segment corresponding to this page identifier is to be deleted, and therefore, the remaining page deletion operations may be performed on the page segment, and after the page segment is deleted upon completion of the page deletion operations, the page identifier is collected.

It should be noted that during fault recovery, disk reading is also in progress. If a page segment includes an index record, the index record may be loaded in the memory for convenient retrieval; and if a page segment includes a key-value pair, a stub record corresponding to the page segment may be retained in the memory to save memory space.

Optionally, after successful fault recovery, the electronic device may reprocess storage requests. The electronic device may determine, among a plurality of storage requests, a first storage request and a second storage request having duplicate LSN numbers. Since the LSN numbers are duplicated, the first storage request and the second storage request are storage

requests with duplicate content, and the electronic device may discard one of the storage requests and process the other storage request.

Optionally, when the electronic device deletes orphan pages corresponding to page identifiers that exist in the page mapping table but are not referenced by the index tree, the electronic device may create a page segment that records the total size of historical invalid data. Each time an orphan page is deleted, the electronic device may accumulate the total size of invalid data in the page segment based on the size of the orphan page.

In this embodiment of the present disclosure, the electronic device may determine the data block count in the disk, and when the data block count is greater than or equal to the preset count, store the page mapping table corresponding to the disk. The electronic device may determine the maximum page identifier in the page mapping table, the third data block that has most recently completed flushing when storage of the page mapping table begins, and the fourth data block on which a data operation is being performed when storage of the page mapping table is completed, and generate the checkpoint information based on the maximum page identifier, the third data block, and the fourth data block, and store the checkpoint information. The electronic device may, when a disk fault is determined, obtain the page mapping table and the checkpoint information, and perform fault recovery processing on the disk based on the page mapping table and the checkpoint information. Since the electronic device can store the page mapping table and the checkpoint information and perform fault recovery processing based on the page mapping table and the checkpoint information, the success rate of fault recovery is improved.

It should be noted that in the present disclosure, the processing steps shown in any of the above embodiments do not constitute a specific limitation on the method. In other embodiments of the present disclosure, more or fewer steps than those in the corresponding embodiment may be included. For example, the data storage process may include some of the steps in the embodiment of FIG. 4, or some steps in the embodiment of FIG. 4 may be replaced by steps having the same function, or some steps in the embodiment of FIG. 4 may be split in a plurality of steps, and so on.

FIG. 14 is a first schematic structural diagram of a data storage apparatus according to an exemplary embodiment of the present disclosure. Referring to FIG. 14, the data storage apparatus 1400 may include: a receiving module 1401, a creating module 1402, a storage module 1403, and a mounting module 1404, where

the receiving module 1401 is configured to receive a data storage request, where the data storage request is used to request storing first data in a disk;

the creating module 1402 is configured to, when a data size of the first data is greater than or equal to a preset threshold, create a first page corresponding to the first data;

the storage module 1403 is configured to store the first data in the first page; and

the mounting module 1404 is configured to mount the first page in an index tree corresponding to the disk.

The data storage apparatus provided in this embodiment of the present disclosure can execute the technical solutions shown in the above method embodiments, and the implementation principles and beneficial effects thereof are similar, which are not repeated herein.

In a possible implementation, the mounting module 1404 is specifically configured to:

determine a first primary page corresponding to the first page in the index tree; and

update index-related information of the first page, update page content of the first primary page, and update index-related information of the first primary page, so as to mount the first page in the index tree.

In a possible implementation, the mounting module 1404 is specifically configured to:

determine a storage address of the first page in the disk and a first key value corresponding to the first data;

generate a first memory record corresponding to the first page, where the first memory record includes the first key value and the storage address of the first page in the disk;

store a first page identifier of the first page in association with a memory address of the first memory record in a page mapping table corresponding to the disk; and

update a state corresponding to the first page to a pending allocation state in the page mapping table, where the pending allocation state is used to indicate that the first page has not been mounted to the index tree.

In a possible implementation, the mounting module 1404 is specifically configured to:

generate a first index record based on the first key value corresponding to the first data and the first page identifier of the first page, where the first index record includes the first key value and the first page identifier; and

add the first index record in the first primary page.

In a possible implementation, the mounting module 1404 is specifically configured to:

generate a first flush-to-disk record corresponding to the first primary page, where the first flush-to-disk record includes a memory address of the first primary page and a storage address of the first primary page in the disk; and

update the memory address of the first primary page to a memory address of the first flush-to-disk record in the page mapping table corresponding to the disk.

The data storage apparatus provided in this embodiment of the present disclosure can execute the technical solutions shown in the above method embodiments, and the implementation principles and beneficial effects thereof are similar, which are not repeated herein.

FIG. 15 is a second schematic structural diagram of a data storage apparatus according to an exemplary embodiment of the present disclosure. Referring to FIG. 15, on the basis of the embodiment shown in FIG. 14, the data storage apparatus 1400 may further include: a processing module 1405 and an update module 1406, where,

the processing module 1405 is configured to perform flush-to-disk processing on the first primary page in the disk based on the first flush-to-disk record;

the update module 1406 is configured to update, in the disk, a state corresponding to the first page in the first primary page to a confirmed allocation state, where the confirmed allocation state is used to indicate that the first page has been mounted to the index tree; and

the update module 1406 is configured to update the state corresponding to the first page to the confirmed allocation state in the page mapping table.

In a possible implementation, the data storage apparatus 1400 may further include: a first determining module 1407 and a deleting module 1408, where,

the receiving module 1401 is further configured to receive a data deletion request, where the data deletion request is used to request deleting second data corresponding to a second key value;

the first determining module 1407 is configured to determine a second page corresponding to the second data in the index tree; and

the deleting module 1408 is configured to delete the second page and index-related information in the index tree, so as to delete the second data.

In a possible implementation, the deleting module 1408 is specifically configured to:

delete the second page;

determine a second primary page corresponding to the second page in the index tree; and

update page content and index-related information of the second primary page.

In a possible implementation, the deleting module 1408 is specifically configured to:

when the second page has not been flushed to the disk, delete the second page and a second memory record corresponding to the second page from a memory corresponding to the disk;

when the second page has been flushed to the disk, delete the second page from both the memory and the disk, and delete the second memory record from the memory; and

delete a memory address of the second memory record and update a state corresponding to the second page to a pending deletion state in the page mapping table corresponding to the disk.

In a possible implementation, the deleting module 1408 is specifically configured to:

determine a second index record corresponding to the second page, where the second index record includes the second key value and a second page identifier of the second page; and

update a state corresponding to the second index record to a confirmed deletion state in the second primary page.

In a possible implementation, the deleting module 1408 is specifically configured to:

generate a second flush-to-disk record corresponding to the second primary page, where the second flush-to-disk record includes a storage address of the second primary page in the disk; and

update a memory address of the second primary page to a memory address of the second flush-to-disk record in a page mapping table corresponding to the disk.

In a possible implementation, the update module 1406 is further configured to:

update a state corresponding to the second page to a confirmed deletion state in the page mapping table.

In a possible implementation, the data storage apparatus 1400 may further include: a first generating module 1409 and a merge module 1410, where,

the receiving module 1401 is further configured to receive a data merge request, where the data merge request is used to request merging a third page and at least one update record of the third page;

the first generating module 1409 is configured to generate a merge record corresponding to the third page based on the data merge request;

the mounting module 1404 is further configured to mount the merge record on the third page, where the merge record is used to mount a next update record of the third page; and

the merge module 1410 is configured to perform merge processing on the third page and the at least one update record to obtain a merged page.

In a possible implementation, the mounting module 1404 is specifically configured to:

determine a second update record among the at least one update record, where the second update record is the last update record among the at least one update record;

add a memory address of the second update record in the merge record; and

update a mapping content corresponding to a third page identifier of the third page to a memory address of the merge record in a page mapping table corresponding to the disk.

In a possible implementation,

the first determining module 1407 is further configured to determine a memory address of the merged page; and

the update module 1406 is further configured to update the memory address of the second update record to the memory address of the merged page in the merge record.

In a possible implementation, the first generating module 1409 is further configured to:

determine a first record count of the at least one update record; and

add the first record count in the merge record.

In a possible implementation, the update module 1406 is further configured to:

update the first record count to 0 in the merge record.

In a possible implementation,

the receiving module 1401 is further configured to receive an update request corresponding to the third page;

the first generating module 1409 is further configured to generate a target update record corresponding to the third page; and

the update module 1406 is further configured to add a memory address of the merge record to the target update record, and update the memory address of the merge record to a memory address of the target update record in a page mapping table corresponding to the disk.

In a possible implementation, the apparatus further includes: a second determining module 1411,

the second determining module 1411 is configured to determine, in the disk, at least one target data block to be subjected to garbage collection, where the target data block includes a plurality of page segments;

the second determining module 1411 is configured to determine a valid page segment in the at least one target data block;

the storage module 1403 is configured to store the valid page segment in a buffer; and

the deleting module 1408 is further configured to delete the at least one target data block from the disk.

In a possible implementation, the second determining module 1411 is specifically configured to:

determine a first data block in the disk, where the first data block is the last data block from a previous garbage collection;

determine a second data block on which a data operation is being performed in the disk; and

determine a data block between the first data block and the second data block in the disk as the at least one target data block.

In a possible implementation, the second determining module 1411 is specifically configured to:

determine a segment state of each page segment in the at least one target data block, where the segment state is a valid state or an invalid state; and

determine a page segment in the valid state among the at least one target data block as a valid page segment.

In a possible implementation, for any page segment, the second determining module 1411 is specifically configured to:

determine a fourth page corresponding to the page segment, where the page segment is obtained after a flush-to-disk is performed on the fourth page;

determine a latest flush-to-disk record corresponding to the fourth page, where the latest flush-to-disk record includes a first storage address corresponding to the fourth page, and the first storage address is the earliest flush address that is currently valid for the fourth page in the disk;

when the storage address of the page segment is greater than or equal to the first storage address, determine the segment state as the valid state; and

when the storage address of the page segment is less than the first storage address, determine the segment state as the invalid state.

In a possible implementation, the data storage apparatus 1400 may further include: a third determining module 1412 and a second generating module 1413, where,

the third determining module 1412 is configured to determine a data block count in the disk;

the storage module 1403 is further configured to, when the data block count is greater than or equal to a preset count, store a page mapping table corresponding to the disk;

the third determining module 1412 is further configured to determine a maximum page identifier in the page mapping table, a third data block that has most recently completed flushing when storage of the page mapping table begins, and a fourth data block on which a data operation is being performed when storage of the page mapping table is completed; and

the second generating module 1413 is configured to generate checkpoint information based on the maximum page identifier, the third data block, and the fourth data block, and store the checkpoint information, where the checkpoint information includes the maximum page identifier, an identifier of the third data block, and an identifier of the fourth data block.

In a possible implementation, the data storage apparatus 1400 may further include: an obtaining module 1414 and a recovery module 1415, where,

the obtaining module 1414 is configured to, when a disk fault is determined, obtain the page mapping table and the checkpoint information; and

the recovery module 1415 is configured to perform fault recovery processing on the disk based on the page mapping table and the checkpoint information.

The data storage apparatus provided in this embodiment of the present disclosure can execute the technical solutions shown in the above method embodiments, and the implementation principles and beneficial effects thereof are similar, which are not repeated herein.

FIG. 16 is a schematic structural diagram of an electronic device according to an exemplary embodiment of the present disclosure. Referring to FIG. 16, the electronic device 1600 may include a processor 1601 and a memory 1602. Illustratively, the processor 1601 and the memory 1602 are connected to each other through a bus 1603.

The memory 1602 stores computer-executable instructions; and

the processor 1601 executes the computer-executable instructions stored in the memory 1602, so as to cause the processor 1601 to execute the method as shown in the above method embodiments.

Correspondingly, an embodiment of the present disclosure provides a computer-readable storage medium, where computer-executable instructions are stored in the computer-readable storage medium, and when the computer-executable instructions are executed by a processor, the method described in the above method embodiments is implemented.

Correspondingly, an embodiment of the present disclosure may further provide a computer program product, including a computer program, and when the computer program is executed by a processor, the method shown in the above method embodiments can be implemented.

Those skilled in the art should understand that the embodiments of the present disclosure may be provided as a method, a system, or a computer program product. Therefore, the present disclosure may adopt the form of an entirely hardware embodiment, an entirely software embodiment, or an embodiment combining software and hardware aspects. Moreover, the present disclosure may adopt the form of a computer program product implemented on one or more computer-usable storage media (including but not limited to disk storage, a CD-ROM, optical storage, etc.) containing a computer-usable program code.

The present disclosure is described with reference to flowcharts and/or block diagrams of the method, apparatus (system), and the computer program product according to embodiments of the present disclosure. It should be understood that each process and/or block in the flowcharts and/or block diagrams, and combinations of processes and/or blocks in the flowcharts and/or block diagrams, may be implemented by computer program instructions. These computer program instructions may be provided to a processor of a general-purpose computer, a special-purpose computer, an embedded processor, or other programmable data processing apparatuses, to produce a machine, such that the instructions executed by the processor of the computer or other programmable data processing apparatuses produce an apparatus for implementing the functions specified in one or more processes of the flowcharts and/or one or more blocks of the block diagrams.

These computer program instructions may also be stored in a computer-readable memory that can direct a computer or other programmable data processing apparatus to function in a particular manner, such that the instructions stored in the computer-readable memory produce an article of manufacture including an instruction apparatus that implements the functions specified in one or more processes of the flowcharts and/or one or more blocks of the block diagrams.

These computer program instructions may also be loaded onto a computer or other programmable data processing apparatuses to cause a series of operational steps to be performed on the computer or other programmable apparatuses to produce a computer-implemented process, such that the instructions executed on the computer or other programmable apparatuses provide steps for implementing the functions specified in one or more processes of the flowcharts and/or one or more blocks of the block diagrams.

In a typical configuration, a computing device includes one or more processors (CPUs), input/output interfaces, network interfaces, and memories.

The memory may include non-permanent storage in a computer-readable medium, a random access memory (RAM), and/or a non-volatile memory such as a read-only memory (ROM) or a flash memory (flash RAM). The memory is an example of a computer-readable medium.

The computer-readable media include permanent and non-permanent, removable and non-removable media that may implement information storage by any method or technology. The information may be computer-readable instructions, data structures, modules of a program, or other data. Examples of computer storage media include, but are not limited to, a phase-change memory (PRAM), a static random access memory (SRAM), a dynamic random access memory (DRAM), other types of random access memories (RAM), a read-only memory (ROM), an electrically erasable programmable read-only memory (EEPROM), a flash memory or other memory technologies, a compact disc read-only memory (CD-ROM), a digital versatile disc (DVD) or other optical storage, magnetic cassettes, a magnetic tape, a magnetic disk storage or other magnetic storage devices, or any other non-transitory medium, which may be used to store information that can be accessed by a computing device. As

defined herein, the computer-readable media do not include transitory computer-readable media (transitory media), such as modulated data signals and carrier waves.

It should further be noted that the terms "include," "comprise," or any other variation thereof are intended to cover non-exclusive inclusion, such that a process, method, article, or apparatus that includes a series of elements includes not only those elements but also other elements not explicitly listed, or also includes elements inherent to such a process, method, article, or apparatus. Without further limitation, an element defined by the phrase "including a ..." does not preclude the presence of additional identical elements in the process, method, article, or apparatus that includes the element.

The foregoing descriptions are merely embodiments of the present disclosure and are not intended to limit the present disclosure. For those skilled in the art, the present disclosure may have various modifications and variations. Any modification, equivalent replacement, improvement, and the like made within the spirit and principle of the present disclosure shall be included within the scope of the claims of the present disclosure.

CLAIMS

1. A data storage method, comprising:

receiving a data storage request, wherein the data storage request is used to request storing first data in a disk;

when a data size of the first data is greater than or equal to a preset threshold, creating a first page corresponding to the first data; and

storing the first data in the first page, and mounting the first page in an index tree corresponding to the disk.

2. The method according to claim 1, wherein the mounting the first page in the index tree corresponding to the disk comprises:

determining a first primary page corresponding to the first page in the index tree; and

updating index-related information of the first page, updating page content of the first primary page, and updating index-related information of the first primary page, so as to mount the first page in the index tree.

3. The method according to claim 2, wherein the updating the index-related information of the first page comprises:

determining a storage address of the first page in the disk and a first key value corresponding to the first data;

generating a first memory record corresponding to the first page, wherein the first memory record comprises the first key value and the storage address of the first page in the disk;

storing a first page identifier of the first page in association with a memory address of the first memory record in a page mapping table corresponding to the disk; and

updating a state corresponding to the first page to a pending allocation state in the page mapping table, wherein the pending allocation state is used to indicate that the first page has not been mounted to the index tree.

4. The method according to claim 2, wherein the updating the page content of the first primary page comprises:

generating a first index record based on the first key value corresponding to the first data and the first page identifier of the first page, wherein the first index record comprises the first key value and the first page identifier; and

adding the first index record in the first primary page.

5. The method according to any one of claims 2 to 4, wherein the updating the index-related information of the first primary page comprises:

generating a first flush-to-disk record corresponding to the first primary page, wherein the first flush-to-disk record comprises a memory address of the first primary page and a storage address of the first primary page in the disk; and

updating the memory address of the first primary page to a memory address of the first flush-to-disk record in the page mapping table corresponding to the disk.

6. The method according to claim 5, further comprising:

performing flush-to-disk processing on the first primary page in the disk based on the first flush-to-disk record;

updating, in the disk, a state corresponding to the first page in the first primary page to a confirmed allocation state, wherein the confirmed allocation state is used to indicate that the first page has been mounted to the index tree; and

updating the state corresponding to the first page to the confirmed allocation state in the page mapping table.

7. The method according to any one of claims 1 to 6, further comprising:

receiving a data deletion request, wherein the data deletion request is used to request deleting second data corresponding to a second key value;

determining a second page corresponding to the second data in the index tree; and

deleting the second page and index-related information in the index tree, so as to delete the second data.

8. The method according to claim 7, wherein the deleting the second page and the index-related information in the index tree comprises:

deleting the second page;

determining a second primary page corresponding to the second page in the index tree; and

updating page content and index-related information of the second primary page.

9. The method according to claim 8, wherein the deleting the second page comprises:

when the second page has not been flushed to the disk, deleting the second page and a second memory record corresponding to the second page from a memory corresponding to the disk;

when the second page has been flushed to the disk, deleting the second page from both the memory and the disk, and deleting the second memory record from the memory; and

deleting a memory address of the second memory record and updating a state corresponding to the second page to a pending deletion state in a page mapping table corresponding to the disk.

10. The method according to claim 8 or 9, wherein the updating the page content of the second primary page comprises:

determining a second index record corresponding to the second page, wherein the second index record comprises the second key value and a second page identifier of the second page; and

updating a state corresponding to the second index record to a confirmed deletion state in the second primary page.

11. The method according to claim 8 or 9, wherein the updating the index-related information of the second primary page comprises:

generating a second flush-to-disk record corresponding to the second primary page, wherein the second flush-to-disk record comprises a storage address of the second primary page in the disk; and

updating a memory address of the second primary page to a memory address of the second flush-to-disk record in a page mapping table corresponding to the disk.

12. The method according to claim 10, further comprising:

updating a state corresponding to the second page to a confirmed deletion state in the page mapping table.

13. The method according to any one of claims 1 to 12, further comprising:

receiving a data merge request, wherein the data merge request is used to request merging a third page and at least one update record of the third page;

generating a merge record corresponding to the third page based on the data merge request, and mounting the merge record on the third page, wherein the merge record is used to mount a next update record of the third page; and

performing merge processing on the third page and the at least one update record to obtain a merged page.

14. The method according to claim 13, wherein the mounting the merge record on the third page comprises:

determining a second update record among the at least one update record, wherein the second update record is the last update record among the at least one update record;

adding a memory address of the second update record in the merge record; and

updating a mapping content corresponding to a third page identifier of the third page to a memory address of the merge record in a page mapping table corresponding to the disk.

15. The method according to claim 14, wherein after performing the merge processing on the third page and the at least one update record to obtain the merged page, the method further comprises:

determining a memory address of the merged page; and

updating the memory address of the second update record to the memory address of the merged page in the merge record.

16. The method according to any one of claims 13 to 15, wherein after generating the merge record corresponding to the third page based on the data merge request, the method further comprises:

determining a first record count of the at least one update record; and

adding the first record count in the merge record.

17. The method according to claim 16, wherein after performing the merge processing on the third page and the at least one update record to obtain the merged page, the method further comprises:

updating the first record count to 0 in the merge record.

18. The method according to any one of claims 13 to 17, further comprising:

receiving an update request corresponding to the third page;

generating a target update record corresponding to the third page; and

adding a memory address of the merge record to the target update record, and updating the memory address of the merge record to a memory address of the target update record in a page mapping table corresponding to the disk.

19. The method according to any one of claims 1 to 18, further comprising:

determining, in the disk, at least one target data block to be subjected to garbage collection, wherein the target data block comprises a plurality of page segments;

determining a valid page segment in the at least one target data block, and storing the valid page segment in a buffer; and

deleting the at least one target data block from the disk.

20. The method according to claim 19, wherein the determining, in the disk, the at least one target data block to be subjected to the garbage collection comprises:

determining a first data block in the disk, wherein the first data block is the last data block from a previous garbage collection;

determining a second data block on which a data operation is being performed in the disk; and

determining a data block between the first data block and the second data block in the disk as the at least one target data block.

21. The method according to claim 19 or 20, wherein the determining the valid page segment in the at least one target data block comprises:

determining a segment state of each page segment in the at least one target data block, wherein the segment state is a valid state or an invalid state; and

determining a page segment in the valid state among the at least one target data block as a valid page segment.

22. The method according to claim 21, wherein for any page segment, determining the segment state of the page segment in the at least one target data block, comprising:

determining a fourth page corresponding to the page segment, wherein the page segment is obtained after a flush-to-disk is performed on the fourth page;

determining a latest flush-to-disk record corresponding to the fourth page, wherein the latest flush-to-disk record comprises a first storage address corresponding to the fourth page, and the first storage address is the earliest flush address that is currently valid for the fourth page in the disk;

when the storage address of the page segment is greater than or equal to the first storage address, determining the segment state as the valid state; and

when the storage address of the page segment is less than the first storage address, determining the segment state as the invalid state.

23. The method according to any one of claims 1 to 22, further comprising:

determining a data block count in the disk;

when the data block count is greater than or equal to a preset count, storing a page mapping table corresponding to the disk;

determining a maximum page identifier in the page mapping table, a third data block that has most recently completed flushing when storage of the page mapping table begins, and a fourth data block on which a data operation is being performed when storage of the page mapping table is completed; and

generating checkpoint information based on the maximum page identifier, the third data block, and the fourth data block, and storing the checkpoint information, wherein the checkpoint information comprises the maximum page identifier, an identifier of the third data block, and an identifier of the fourth data block.

24. The method according to claim 23, further comprising:

when a disk fault is determined, obtaining the page mapping table and the checkpoint information; and

performing fault recovery processing on the disk based on the page mapping table and the checkpoint information.

25. A data storage apparatus, comprising: a receiving module, a creating module, a storage module, and a mounting module, wherein:

the receiving module is configured to receive a data storage request, wherein the data storage request is used to request storing first data in a disk;

the creating module is configured to, when a data size of the first data is greater than or equal to a preset threshold, create a first page corresponding to the first data;

the storage module is configured to store the first data in the first page; and

the mounting module is configured to mount the first page in an index tree corresponding to the disk.

26. An electronic device, comprising:

at least one processor; and

a memory connected to the at least one processor in a communication way;

wherein the memory stores instructions executable by the at least one processor, and the instructions, when executed by the at least one processor, cause the electronic device to perform the method according to any one of claims 1 to 24.

27. A computer-readable storage medium, wherein computer-executable instructions are stored in the computer-readable storage medium, and when the computer-executable instructions are executed by a processor, the method according to any one of claims 1 to 24 is implemented.

28. A computer program product, comprising a computer program, wherein when the computer program is executed by a processor, the method according to any one of claims 1 to 24 is implemented.